17-06-2026

Software Licence Selection and Management in GÉANT – 2026 Update

Grant Agreement No.:	101194278
Work Package:	WP9
Task Item:	T2
Nature of Document:	Guide
Dissemination Level:	PU
Lead Partner:	UoB/AMRES
Document ID:	GN5-2-26-2B18C8
Authors:	Branko Marović (UoB/AMRES); Magdalena Rząca (GÉANT Association)
Licences:	This work is licensed under CC BY 4.0 [1]

Abstract

This document provides a detailed guide to software licence selection, declaration, compliance, and management in GÉANT. Intended for software development teams, it contains both essential information and practical step-by-step instructions. It is divided into two main parts: Essential Aspects and Elements of Software Licensing and Complying with a Selected Licence.

Contents

Figures

1 Introduction

The primary objective of this guide is to explain the intricacies of licence selection, declaration, compliance, and related tasks, offering a step-by-step breakdown of licensing mechanisms for software development teams.

This guide is divided into two main parts:

1. The first part, Essential Aspects and Elements of Software Licensing, provides developers with key insights into software licence selection and management, outlining the tasks they must perform, and the procedures required to engage with services that support licensing. This part provides a straightforward overview of the elements necessary for efficient preparation, information gathering, and software licensing.
2. The second part, Complying with a Selected Licence, explains how to implement the selected licence from a developer's perspective, providing instructions that facilitate the licensing process. It gives in-depth information, along with simple but vital guidance on creating essential licensing artefacts.

This document does not detail individual open source software (OSS) licences, software composition analysis (SCA) tool usage, licence compatibility, or the remediation of licence conflicts. This is intentional, as developers may not need to deal in depth with these complex tasks because they can rely on support from the software licensing team. Where such details are necessary, they are covered in separate guides provided by the software licensing team.

For comprehensive information on licences, compatibility, and licence selection, consult the training and reference materials listed in the Resources section of this guide. Access to certain references in this guide may be restricted to GÉANT project participants or require authentication via eduGAIN [2] or other supported credentials.

2 Essential Aspects and Elements of Software Licensing

This part of the guide covers the following topics:

- Open source software and licensing in GÉANT.
- OSS obligations and licence compatibility.
- GÉANT IPR policy and licence governance.
- SCA and SLA tools, including Mend.
- Licensing process, decisions, and artefacts.
- Licensing options and tracking for software, documentation, data, and other works.

2.1 OSS Benefits and Licensing Considerations

2.1.1 Benefits of OSS

OSS is extensively used in technology products, services, business, governments, research, and education. It provides and supports affordability, transparency, collaboration, and technical innovation. It empowers individuals and organisations to take control of their software solutions, adapt them to their needs, and contribute to a global community of developers and users.

Key benefits include:

1. **Cost-effectiveness** – OSS is often free to use, significantly reducing software acquisition and licensing costs for individuals, businesses, and organisations. This is especially critical for smaller businesses, educational institutions, and governments with budget constraints.
2. **Transparency** – OSS is built on open and transparent development processes. Anyone can review the source code to understand how the software works, enhancing trust and security. This is particularly important for software used in critical applications, such as cybersecurity and healthcare.
3. **Community collaboration** – OSS projects often have large and diverse communities of developers and users who collaborate to improve the software. This leads to rapid bug fixes, updates, and feature enhancements, fostering innovation and knowledge sharing.
4. **Customisation** – Users can modify and customise the code to meet specific needs. This flexibility allows organisations and individuals to adapt software to their unique requirements, offering them a competitive advantage.
5. **Vendor independence** – Unlike proprietary software, where users are often locked into a vendor's ecosystem, OSS offers greater freedom and flexibility, as users have access to the source code and can switch providers or modify software as needed, including integration with other products.
6. **Longevity** – While proprietary software may be discontinued by the vendor, leaving users without support, OSS tends to have a longer lifespan. Communities can continue maintaining and developing the software if the original team slows down or abandons it.

7. **Security** – Although OSS is not immune to vulnerabilities, its transparency allows a global community to audit the code continually. When vulnerabilities are discovered, they can be fixed quickly. Proprietary software security, conversely, depends on the vendor's resources and priorities.
8. **Education and learning** – OSS encourages learning and skill development. Students and aspiring developers can study, modify, and contribute to OSS projects, gaining practical experience in real-world software development.
9. **Compatibility** – Open standards and OSS promote compatibility and interoperability between different software, reducing barriers to data exchange and collaboration.
10. **Ethical values** – The OSS movement is rooted in values such as transparency, collaboration, and the belief that software should be a public good. Many individuals and organisations align with these values.
11. **Global accessibility** – OSS is accessible to users worldwide, regardless of location or economic status, promoting digital inclusion and levelling the playing field.

2.1.2 OSS Licensing in R&E

R&E increasingly relies on OSS. Developers, National Research and Education Networks (NRENs), and GÉANT all use, adapt, create, and endorse OSS.

The ICT and R&E communities value OSS for its cost-effectiveness, transparency, customisability, vendor independence, longevity, security, compatibility, and accessibility, as well as for its collaborative nature and ethical, educational, and societal benefits.

Key factors related to the licensing of OSS in the R&E sector include the following:

- OSS licences ensure that software remains free, prevent appropriation, and reduce the risk of abandonment.
- Declaring a software licence simplifies the selection of code and libraries for use in a project, and facilitates usage, adaptation, and contribution by other developers.
- Licensing is critical when distributing or sharing software, as OSS licences impose specific conditions.
- Licence declaration and compliance are essential for legal and usability reasons, increasing transparency and fostering collaboration.
- Adhering to applicable licences (including those of dependencies) enhances the visibility and credibility of a software project within the wider community.

2.2 GÉANT IPR Policy and Licensing Framework

2.2.1 Overview of GÉANT IPR Policy

The GÉANT IPR Policy [3] applies to intellectual property (IP) generated within the GÉANT project, including open source software (OSS). It provides recommendations and rules, which this document translates into practical and actionable instructions. The policy – which is available at [3] – aims to establish a framework for IP generated by the GÉANT project, applying to all GÉANT participants. It offers practical guidance on IPR and seeks to establish a cooperative approach to IP protection and fair use in GÉANT projects. The IPR Policy also upholds the principles of findability, accessibility, interoperability, and reusability (FAIR) for the use of GÉANT project IP. Following approval by the General Assembly, it has been binding since January 2023.

In the context of the GÉANT project, specific requirements and recommendations for software projects are set out in its IPR Policy [3]:

1. **Establish clear naming and branding** and define how the project should be **packaged into products and repositories**. Ideally, OSS should be hosted in a public versioned repository, with GÉANT GitLab as the preferred platform [4][5]. The relationship between packages and their components will likely influence licensing decisions.
2. Every GÉANT software project should select and **apply an OSS licence** that meets the needs of both the development team and the user community.
3. **Start the licensing process early** to streamline licensing and compliance.
4. The **chosen licence must be compatible** with all used components' licences, to eliminate IPR and licensing risks for GÉANT.
5. It is preferable to place the OSS source code in a public and **versioned code repository,** with a **clear licence indication**.
6. **Copyright** information must acknowledge GÉANT's involvement and support, underscoring that the work was conducted within the GÉANT project or received support from it. The authors of the produced software should also be identified.
7. **Assess components and software** using common software quality and trustworthiness checklists to ensure quality and reliability. Examples include *TinyMCE – Open Source Software Evaluation Checklist* [6], *Red Hat – Checklist for Measuring the Health of an Open Source Project* [7], and *EURISE Network Technical Reference – Software Quality Checklist* [8].
8. **Use GÉANT software composition and licence analysis (SCA and SLA) services** to conduct related reviews and audits, designed to determine the appropriate OSS licence and ensure compliance. Identifying and addressing software vulnerabilities through SCA improves quality and benefits the broader community.
9. **Set up contribution, communication, and governance workflows** to ensure compliance with the software's licence.
10. **Adhere to the standards of the domain community** in software development, licensing, software metadata, documentation, registration in relevant community registries, citation, and promotion.
11. If applicable, **enable and advise on the citation and referencing** of software in scientific papers, presentations, and tutorials, ensuring clear and persistent references.

These, along with further recommendations for various software project stakeholders, are detailed in the GN5-1 *Software Licence Selection and Management* guide [9], *Deliverable GN5-1 D9.4 Open Source and Licence Support Report* [10] and *Deliverable GN5-2 D9.5 Open Source and Licence Support Report* [11].

From a software development perspective, undertaking software composition analysis (SCA) using a tool that scans components to identify their licences is crucial for determining which licence to apply. Non-compliance with OSS licence terms can have serious legal and financial consequences, so due diligence is essential. The IPR Policy stresses the importance of IP protection, introduces the GÉANT IP Register to document project results, and highlights the need for scanning code with an SCA tool to ensure licence compliance. It also underlines the role of the IPR Coordinator in supporting development teams through the licence selection process.

To summarise the IPR Policy:

- **All OSS licences** [12] [13] **are permitted**.
- The IPR Policy strongly recommends the use of permissive licences.
- Copyleft licences (weak, strong, or network-protective) can be applied as needed, in consultation with the IPR Coordinator.

- The OSS code must be scanned to ensure licence compliance and a licence chosen and declared with the OSS code.
- The IPR Coordinator provides final recommendations and maintains the GÉANT IP Register.

2.2.2 Roles and Responsibilities

This summarises the roles and responsibilities defined by the GÉANT IPR Policy [3], and the established operational arrangements for managing IP and ensuring OSS licence compliance across the project.

- **GÉANT General Assembly** – Holds ultimate strategic and decision-making authority over the IPR Policy. It approves the Policy and any amendments, authorises GÉANT's exploitation of Project IP, and acts as the final decision-maker for disputes not resolved through mediation within two months. It also provides the mandate for enforcement, ensuring alignment with IP protection and collaborative objectives.
- **GÉANT Organisation** – Where authorised by the General Assembly, holds non-exclusive, perpetual, irrevocable, worldwide rights to exploit Project IP, including sublicensing. It ensures the structures and processes required to implement the IPR Policy are in place. It does not warrant the validity of IP rights and is not liable for related claims; Partners and Participants must provide appropriate indemnities.
- **IPR Coordinator** – Appointed by GÉANT, the IPR Coordinator is the operational lead and subject matter expert for all IPR matters. Responsibilities include promoting IP awareness, maintaining the IP Register, advising Partners and Participants, managing licence documentation, conducting OSS compliance reviews, authorising audits, mediating disputes, and updating the Policy bi-annually. The role ensures consistent identification, protection, and sharing of IP across projects.
- **IPR Committee** – An ad hoc body convened by the IPR Coordinator with relevant stakeholders. It assesses whether Project IP should be protected and under what terms, including licensing options such as exclusive, non-exclusive, royalty-free, or donation, and may recommend protection even where a Partner opts not to pursue it.
- **Partner Organisations** – Own the Results they create and carry primary responsibility for protecting Project IP. They must report generated IP to the IPR Coordinator, ensure appropriate agreements with Participants, grant GÉANT non-exclusive exploitation rights free of charge, and may license third parties where this does not conflict with the Policy or Agreement. They must also ensure that Participants understand and comply with IPR obligations from the outset.
- **Work Package Leaders, Task Leaders, and Team Leads** – Responsible for implementing and enforcing the IPR Policy within their scope. They oversee OSS licence compliance and Software Composition Analysis, ensure remediation actions are completed, schedule follow-up vulnerability scans, and confirm that all OSS outputs are recorded in the IP Register before Public Disclosure. They must identify IP early, report it promptly, manage contributions and dependencies, prevent licence conflicts, ensure compliant use of third-party software, and coordinate closely with the IPR Coordinator during reviews and audits.
- **Participants** – Any individual or external contributor engaged by a Partner, including GÉANT staff and third parties. They must report any IP they create to their Partner, disclose Background IP and known risks before starting work, and maintain confidentiality of Trade Secrets. They must not disclose Project outputs without prior approval. Where external, they must operate under written agreements covering IP ownership or transfer, confidentiality, indemnification, and confirmation that any sideground IP does not infringe third-party rights.

Practical licence governance in GÉANT is **led by the IPR Coordinator** and **supported by WP9 Task 2's Software & Licence Management (SLM) subtask**, i.e. the software licensing team.

The SLM team in WP9 T2:

- Assists those wishing to understand and manage OSS licences, master the tools provided, and apply them.
- Provides knowledge and support to solution designers, developers, and skilled promoters on licences and IPR.
- Offers technical and implementation support on open source software and licence management through two services:
  - **Software composition analysis (SCA)** – Technical and practical assistance for development teams in managing software components and their licences.
  - **Software licence analysis (SLA)** – Assistance for software teams in aligning project and licensing decisions with the GÉANT IPR Policy and the guidance provided by the IPR Coordinator.

2.3 OSS Licensing Obligations and Compatibility

2.3.1 OSS Licence Types and Conditions

Selecting a licence is not straightforward and is a task most developers prefer to avoid. While the *GÉANT IPR Policy* [3] favours permissive licences and names examples such as MIT, BSD, and EUPL, the available options are often constrained by the licences of core components or the frameworks that the software relies on. It is therefore sensible for developers to explore specific licences, their implications, and compatibility only when the constraints and candidate licences are known within the licensing process, supported by the WP9 licensing team and the GÉANT IPR Coordinator.

For example, a developer might have to use the Apache licence if the framework they are using mandates it. Alternatively, if there are unavoidable GPL or AGPL dependencies, they may need to opt for a compatible licence, typically the most restrictive of the applicable licences. Conversely, in cases where no pre-existing limitations exist, such as with a new project or where all essential components use permissive licences, the development team can select one or more candidate licences based on key factors such as code-sharing and modification provisions (with copyleft licences being more demanding), the ability to relicense (sometimes incorrectly referred to as sublicensing, which the team may wish to prohibit or allow), and the handling of patents (waived, protected, or not addressed). (For further information about software selection, see Section 2.5 Licence Selection and Management Tools and Services.)

Various licence conditions are often classified as:

- Rights/permissions – what you are allowed to do.
- Requirements/obligations – mandatory compliance artefacts, such as copyright, disclaimers, licence text, notices, relevant source code, build and installation instructions, etc.
- Restrictions/limitations – what you are prohibited from doing.
- Other characteristics – typical uses, classification, compatibility, legal features, origin, community, endorsement, and more.

These features are further outlined in models such as the EC's Licensing Assistant tool [14], which, when suitable licensing options exist (see 2.8 Licence Options and Multi-Licensing), greatly facilitates the selection of the most suitable licence aligned with the developers' strategy and user community. Sites such as choosalicense.com [15] can also help guide developers in choosing the right open source licence for their requirements.

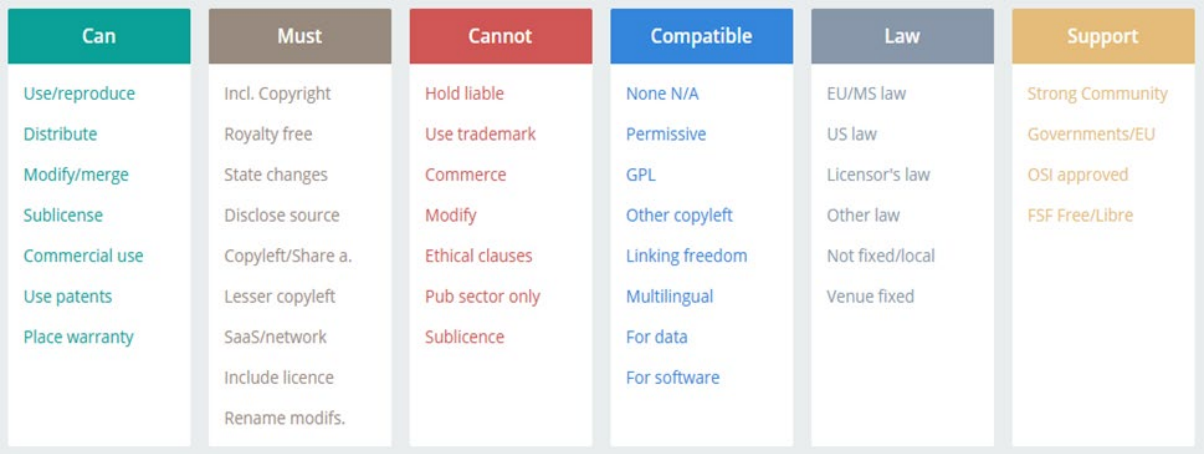

Figure 2.1: OSS conditions from Licensing Assistant – *Find and Compare Software Licenses* [14]

2.3.2 Compatibility of Frequently Used Licences

There are two common interpretations of licence compatibility. A less restrictive, more often applied, and symmetrical type of compatibility indicates that components with distinct licences can be used in the same project. This may be achieved by relicensing one or both components, or by selecting a third licence for the encompassing product. A more restrictive and direct, yet asymmetrical, interpretation determines whether a component under one licence may be used in software governed by another licence.

Regardless of the interpretation, compliant types of "use" can sometimes be achieved by altering the system architecture. For example, a problematic component may be extracted into a standalone service and thus a separate project, allowing its integration without the need to change the software licence. Only a limited set of licences is frequently used in GÉANT, and only a few of these are common sources of compatibility issues. Figure 2.2 provides an orientational diagram describing the relationships and compatibility of these licences.

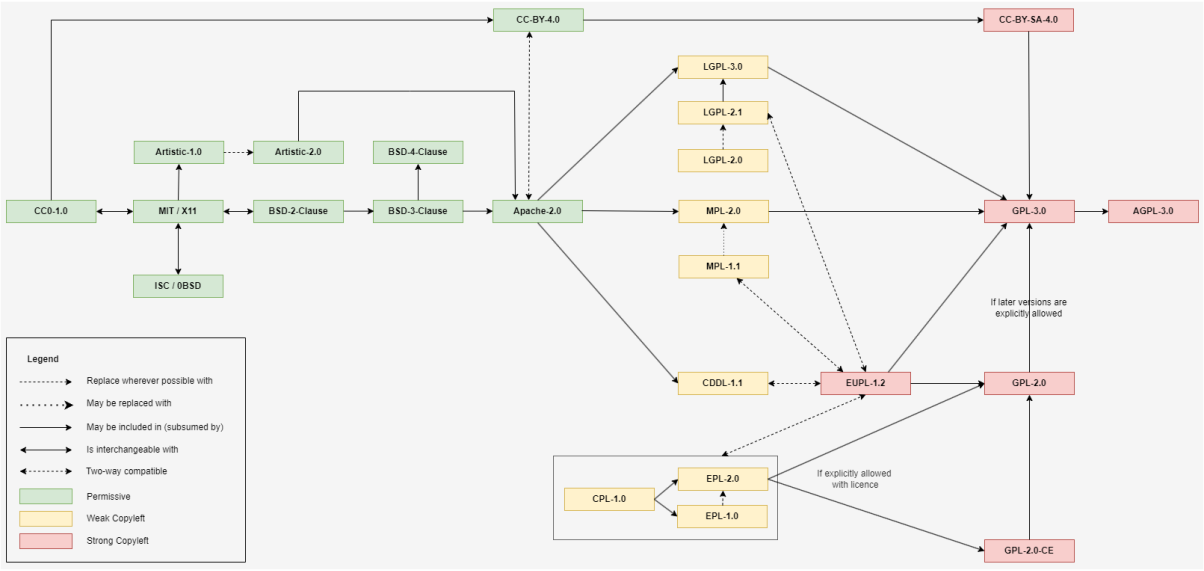

Figure 2.2: Relationships between OSS licences commonly used in GÉANT projects

WP9 has compiled comprehensive information about OSS licences, some of which is also available through Mend, the tool used for SCA (described in Section 2.8). OSS licences are additionally described in several documents produced by the licensing team: *Reference Information about OSS Licences and Tools* [16], *OSS Licences and Licence Selection* [17], and *Open Source Licences Used in GÉANT* [18].

2.3.3 Obligations under GPL: Special Considerations

Among open source licences, the GNU General Public License (**GPL**) family imposes distinct obligations. As **copyleft** licences, they require derivative works or redistributed versions to adopt the same GPL terms. Distributing software that incorporates GPL-licensed code legally obligates you to **release the source code publicly** under identical conditions. While this aligns with open science ideals of transparency and shared knowledge, compliance demands deliberate planning, especially in collaborative or commercial settings.

2.4 Licence Governance in GÉANT

2.4.1 Governance Structure and Compliance Support

The goal of licence governance in GÉANT is to ensure compliance with GÉANT's IPR Policy while respecting the licences of dependencies and domain community standards. The WP9 SLM team provides services, guidance, and support in these areas, including through dedicated SCA and SLA services.

These services, described in more detail in sections 2.7 and 2.9, complement other software review services provided by WP9 Task 2 Software Governance and Support, such as SonarQube Setup Assistance and Extended Source Code Review [19]. Through SCA and SLA services, the licensing team ensures that key licensing concerns are addressed by:

- Assessing the licences of components and prior IP.
- Selecting an open source licence that meets the project's needs.
- Ensuring the selected licence is compatible with the components' licences.
- Ensuring compliance with the chosen licence.

The licensing team has implemented robust processes for managing dependencies and licences and achieving compliance with the *GÉANT IPR Policy* to support GÉANT software teams with IPR and licensing issues. It provides access to expert tools for analysing and managing open source licences. SLM has collaborated with many GÉANT software teams to assess their licensing situations and decisions, offering recommendations to support reliable and effective IPR management in line with the *GÉANT IPR Policy*.

GÉANT development and maintenance teams can contact SLM through the GÉANT Slack channel or by email. SCA and SLA services are requested via a software review request submitted to the GÉANT Jira Software Tools Help Desk [20], which also tracks their progress. Several iterations of analysis and adjustments to licences or dependencies may be required to achieve satisfactory IPR status. The IPR Coordinator is available to assist with licensing decisions.

SCA and SLA, along with GÉANT's software licensing certificates [21], offer valuable opportunities to align software projects with GÉANT's and external licensing and policy expectations, thereby promoting the reuse of OSS developed within the GÉANT project and strengthening the project's broader impact. Analysis, selection, and validation of software licences may lead to changes that provide positive indicators for GÉANT and potential external users and contributors. Furthermore, both services require registering and publishing software using

GÉANT's internal software tools, some of which provide public visibility. This may be particularly significant for smaller developments with potential broader applicability.

Engaging with a structured software licensing process moreover helps development teams to evaluate project components and licences while considering aspects related to authorship, ownership, external relations, and documentation artefacts, resulting in improved standardisation in these areas. Licensing reviews may also bring in other individuals, such as new team members, providing them with an opportunity to familiarise themselves with the software. As an additional benefit, analyses conducted for add-on modules for externally developed open source platforms assessed for use in GÉANT may contribute to the broader software communities of these platforms, by providing feedback on their overall licensing status and component security.

The process of addressing issues through licensing analysis, together with the resulting reports and decisions, can also provide valuable insights for evaluating software solutions and services at GÉANT Product Lifecycle Management (PLM) gates [22].

The licensing team produces several guides to assist development teams with licensing and related services. An overview of OSS licences and their selection can be found in the *OSS Licences and Licence Selection* guide [17]. It is recommended that this guide be read first, particularly before requesting the SLA service, as this requires the involvement of the software team in licence selection. This guide additionally provides information on interpreting software composition analysis results.

However, if you are already familiar with OSS licences, or are seeking summaries of licences commonly found in GÉANT projects or that typically present licence compatibility issues, you may want to proceed directly to the *Open Source Licences Used in GÉANT* guide [18].

All relevant materials developed by the team are listed in Section 4.3 *Further Reading*, while training and webinar materials are listed in Section 4.2.

2.4.2 Declaring a Licence in Code Repositories

Declaring an open source licence when publishing code in a public repository is essential for legal clarity, responsible reuse, and alignment with community and funder expectations. Legally, **code shared without an explicit licence** defaults to "all rights reserved", prohibiting others from using, modifying, or redistributing it, even when publicly accessible. Applying a well-defined licence ensures compliance with legal requirements while empowering others to reuse the work. Specifying a licence communicates the terms of use, protecting both the **developer's rights** and **users' obligations**, including attribution requirements, disclaimers, and redistribution conditions. This is particularly crucial in collaborative environments and for publicly funded projects, where compliance, reuse potential, and transparency are essential values.

2.4.3 FAIR Principles and Licensing Implications

Licensing also plays a pivotal role in upholding the **FAIR principles (Findable, Accessible, Interoperable, and Reusable)**, widely endorsed in research and open science. Licences make software more **Findable** in a variety of ways, including by making it available in code repositories and facilitating searching, increasing its eligibility for catalogues and signalling openness for its reuse thereby increasing its visibility. The **Reusability** of digital assets, including software, hinges on a clear licence: Without one, code cannot be integrated into other projects, workflows, or datasets, undermining reproducibility, and collaboration. A declared licence further supports **Accessibility** by resolving legal ambiguity and enhances **Interoperability**, particularly when integrating projects with differing licensing terms.

2.5 Licence Selection and Management Tools and Services

2.5.1 Software Composition Analysis (SCA) Service

This service assists development teams by setting up a project within an SCA tool and providing insights into external components. It is suitable for both **one-time software analysis** and **continuous monitoring**, identifying third-party components, their licences, and potential IPR infringements or security vulnerabilities. The service can be combined with other software review services or performed independently. Repeated analyses can determine how changes in software and dependencies affect licence compliance and identify new or pending vulnerabilities. For ongoing monitoring, the analysis setup can be integrated into the project's continuous integration (CI) platform.

The SCA is currently based on Mend (described in more detail in Section 2.8), which identifies third-party components in projects and gathers information about their licences and security vulnerabilities. Mend uses a comprehensive database to address two critical aspects:

- Analysing components and their licences to reduce the risk of IPR infringement, which could have significant financial consequences, by supporting licence compatibility and compliance.
- Reporting security vulnerabilities, which complements SonarQube and extended code reviews.

The licensing team sets up the project in the Mend SCA tool, which generates reports on software composition and potential deviations from established policies. The visibility of reports and the created Mend project can be configured during project setup or adjusted after results are obtained. The primary report, the "Risk Report", details the software composition, including components, their licences, and related risks and vulnerabilities.

The designated leader or expert from the development team receives this report and supports its interpretation, although the development team should be able to interpret it independently. The licensing team assists where needed, and developers can request additional feedback on risks related to licences and IPR infringements.

A summary of the SCA service is available in *Software Reviews* [19] and *Software Composition Analysis* [23], with more details provided in the *Client Guide for Software Composition Analysis (SCA)* [24].

To request a software composition analysis, email sw-licences@software.geant.org**,** post in the **#sw-licences** channel on the GÉANT Project Slack [25] or submit a Software Review Request via the Help Desk [20].

2.5.2 Mend SCA Tool

Mend [26] is an online tool that GÉANT currently uses for open source licence and security compliance. The tool was implemented for the GÉANT Project and is technically supported by the WP9 Task 2 SLM team. It **detects software components**, **their licences, and vulnerabilities**, and can be integrated with development environments and CI pipelines to identify libraries with security or compliance issues. It reports severe bugs, problematic licences, new versions, and available fixes, simplifying open source management.

Mend creates a **dependency inventory** matched against its database, providing **licence details**, warnings about outdated or risky libraries, and vulnerability data. Licence information includes type, risk level, patent handling, summaries, and excerpts from original texts.

Mend streamlines and partly automates IPR compliance verification, improving visibility and control over open source risks. The licensing team configures and maintains the tool for analysis, including user permissions, project visibility, and approved or rejected libraries and licences, if provided by the development team. Usage guidance is available in the *Mend Short Guide for End Users* [27].

Mend analyses projects in several ways. Code may be stored locally, and scans triggered manually to assess recent changes – see *Adding a Project to Mend (Scan Flow)* [28]. Software can be scanned and presented as one project or per product. As Mend does not handle versioning, each software version is scanned as a separate project.

The tool scans directories to detect components and identify vulnerabilities, licence conflicts, or risks, checking digital signatures and component names against its database. Results appear in the Mend web interface, facilitating assessment of a GÉANT product for compliance with the IPR Policy without manual code review. Scans populate the dashboard and support various compliance reports.

The web interface provides multiple panels for reviewing and analysing scans as projects or products, offering an overview of code status with links to details of related libraries, products, and licences. Dashboard segments lead to detailed pages and reports with charts and tables. The dashboard presents:

- **Product Alerts** – outdated libraries and latest versions.
- **Security and Quality** – numbers of vulnerable libraries by severity, libraries with vulnerabilities and newer versions, and the most vulnerable or buggy libraries.
- **Libraries** – details on libraries, licences, and their use per product or project.
- **Licence Analysis** – licence types and usage statistics.

Additional licence information is available through reports. The Risk Report summarises key aspects of libraries, licences, security, and quality, displaying risk-related data through panels and tables. Security and licence details also appear elsewhere, for example, in the Product Dashboard.

Mend's reports rely on an internal, continuously updated database of libraries, versions and vulnerabilities, licences, and licence conflicts, so results may change over time even without new scans.

Mend provides insights into OSS licence types, copyright, patent, and royalty terms, linking requirements, and compliance with open source norms. Its experts have analysed numerous licences and assigned risk scores to help developers assess associated risks.

The main metric used in Mend, the Copyright Risk Score [29], measures the degree of loss of exclusive control when using code under a given licence. This is more relevant for organisations evaluating commercial risk than for projects releasing open source code. Lower scores (green) correspond to permissive licences, while higher scores (red) indicate strong copyleft. Licences are classified by copyleft level (none, partial, or full) and linking type (non-viral, dynamic, or viral). A Patent and Royalty Risk Score indicates whether a licence is royalty-free.

Mend integrates with development environments and build tools and can be incorporated into CI pipelines to trigger scans on each commit in GitHub [30], GÉANT GitLab [4][5], and Bitbucket [31]. Integration with Bamboo [32] CI/CD software is described in *Automated Mend Scans with Bamboo* [33].

Originally designed for commercial IP risk assessment, Mend is being adapted by its vendor for OSS licence compatibility checks. It may not always detect licences accurately, especially for multi-licensed or ambiguously licensed components, and sometimes identifies only indicative ("suspect") licences requiring manual verification. The tool also lacks support for software versioning and differential reports. These limitations hinder fully automated monitoring of components and licences, but Mend remains more efficient than manual analysis. Final licence verification is nevertheless carried out through the Software Licence Analysis (SLA) service (see 2.5.3), as human judgement is still required for this.

The WP9 T2 SLM team and IPR Coordinator are currently reevaluating different SCA tools to verify whether Mend is still the best choice for GÉANT's SLA Service.

2.5.3 Software Licence Analysis (SLA) Service

In addition to ensuring legal compliance, selecting the appropriate open source licence for a software is a key element in fostering a collaborative and transparent development environment, as it formally defines the terms for sharing, modifying, and distributing the software. This selection process considers project objectives, developer preferences, the desired level of collaboration, the software's ecosystem, licences of related products, and users' expectations, as well as the constraints imposed by the licences of dependencies and other background and sideground IP. Choosing a licence also involves assessing the effort required to resolve licence compatibility issues and ensuring compliance with legal, regulatory, and funding requirements.

Before requesting an SLA, it is recommended that you tentatively select a standard OSI-approved licence that aligns with your goals and needs, using tools such as the Licensing Assistant tool [14] and the choosealicense.com [15] guide to support your decision. This exercise helps identify the most suitable licence. However, available options may be limited by compatibility requirements with the licences of third-party components used in the project.

If your project includes an SBOM file and you want to perform a preliminary licence analysis, you can also use the emerging OSS tool Hermine [34] that can be integrated into CI/CD pipelines, to interpret component licences and convert them into concrete legal obligations aligned with an open source policy. Based on its knowledge base and accurate component metadata from SBOM, this tool could, to some extent, automate licence compatibility checks, suggest a suitable project licence for different usage contexts (distribution, network exposure, internal use), and draft legal notices.

Often only the most restrictive licence will be compatible with all components of a software. However, this is not always the case, especially when multiple permissive licences are involved. Furthermore, sometimes the most suitable licence that is compatible with most or all the components may not necessarily be among those used by components. These compatibility requirements are discussed in more detail in Section 2.3.1.

Software Licence Analysis addresses these and other complexities, providing technical consultancy; offering a comprehensive understanding of third-party libraries used in a software project, their licences, and how they relate to the project; discussing the software's licensing; and giving recommendations and validation of project documentation artefacts. This understanding is crucial for selecting the appropriate licence and ensuring compatibility between all licences involved. The service is highly recommended for software development teams seeking to validate third-party licences, establish or review their project's licence, verify compliance with both the licence and the *GÉANT IPR Policy*, or assess the impact of potential licence changes, including changes to dependencies' licences.

The service builds on the results of prior software composition analysis (SCA), with manual checks of libraries where necessary. It includes customisation of the SCA tool's licence settings, project licence selection based on an analysis of dependencies, checking alignment with licence requirements and the *GÉANT IPR Policy*, and a review of related documentation artefacts. If the SCA tool is integrated into a Continuous Integration (CI) pipeline, the service team will work with you to customise the relevant settings.

A summary of the SLA service is available in *Software Reviews* [19] and *Software Licence Analysis* [35].

GÉANT Project participants can request a software licence analysis by posting in the **#sw-licences** channel on the GÉANT Project Slack [25], or submitting a Software Review Request via the Help Desk [20].

2.5.4 Other Tools

A comprehensive list of licensing services and tools – both internal to the project and public – is provided in section 4.4. A regularly updated list of OSS licensing tools and other resources is available at [16], which also covers some alternative tools for software composition and licence analysis [36].

Publicly available tools to support licence selection include the above-mentioned Licensing Assistant tool [14] and the choosealicense.com [15] site.

2.6 Licensing Process

The typical steps for managing licences are:

1. Gather information (can be done using Mend).
2. Document (can be partially done using Mend).
3. Remediate dependency issues.
4. Create licence-related artefacts (to ensure compliance) and declare the licence in the software's repository.

These steps are preceded by preparatory activities and decisions and should be followed by ongoing measures to ensure continuous licence management. Detailed information on the preparatory steps, the process itself, and long-term licence management activities within GÉANT is provided in the following sections. For further information, refer to GÉANT's Open Source Licensing and Compliance training [37], *GN5-1 Deliverable D9.4 Open Source and Licence Support Report* [10], and *Deliverable GN5-2 D9.5 Open Source and Licence Support Report* [11].

2.6.1 Licensing Workflow Overview

The licensing workflow begins with a SCA request, which may be accompanied by a request for an SLA. If no significant issues arise and the current or proposed licence is clear and agreed with the IPR Coordinator, further analysis may be shortened or skipped. If the SLA is not requested, developers may make their own changes based on the SCA results before deciding they are ready for the final SCA, licence selection, and implementation. Changes made during remediation or a significantly postponed SLA may necessitate a repeated SCA to validate changes and facilitate the resolution of remaining issues.

The licence implemented during the creation of licence-related artefacts must still be approved by the IPR Coordinator. This stage involves reviewing the necessary licence- and copyright-related files, with developers updating these files according to the provided guidance. At the end of the process, software developers complete a survey to provide feedback on all the steps they have taken.

Continuous licence management is carried out by the development team based on their SCA–SLA experience, with advice from the licensing team. The licensing team may adjust project configuration in Mend and configure it for the chosen allowed or prohibited licences. While the development team should perform further monitoring and adjustments after successful licensing, it is recommended to repeat the SCA–SLA process following any significant changes to ensure the project remains compliant.

An overview of the GÉANT OSS licence management workflow is shown in Figure 2.3 below.

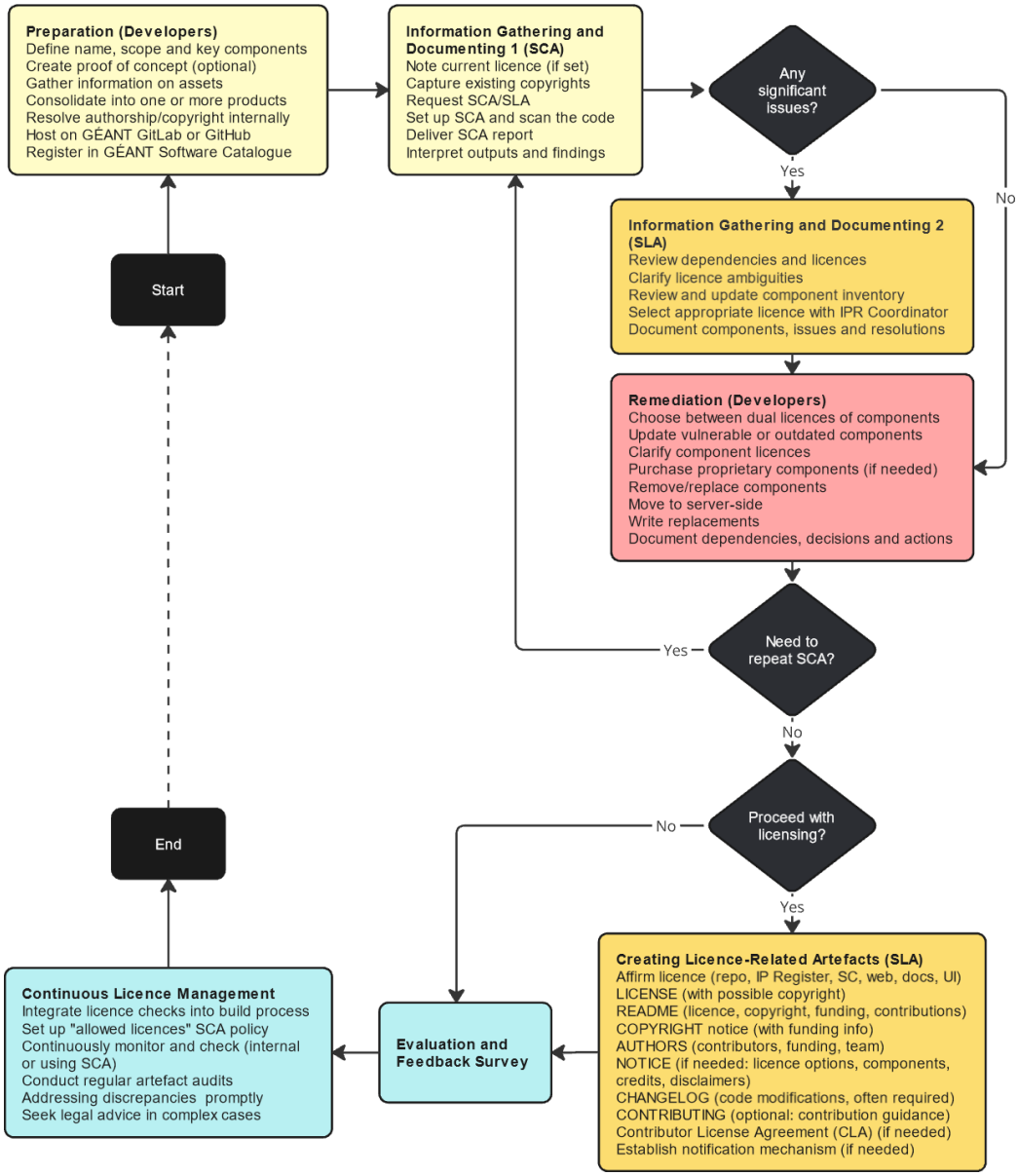

Figure 2.3: Overview of the GÉANT OSS licence management workflow

Each step in the workflow is described in more detail in the following sections.

2.6.2 Preparation

- Decide on the software name, **grouping of subprojects**, and the use of available contributions.
- New projects may require a proof of concept or **prototype** to identify and validate key components.
- Gather **pre-existing information** and documentation for components, models, assets, and relevant specifications and standards.
- Consolidate project components into a **single repository project** or **clarify their relationships** if they should remain **separate**.
- Address authorship and copyright matters internally.
- Ensure the software project is hosted on GÉANT GitLab [4][5] or GitHub [30].

- Register the software project in the GÉANT Software Catalogue [38].

All components of a closely interconnected software development project should reside in a single repository, ideally on GÉANT GitLab. However, some developers may opt for GitHub or mirror their GitLab project there to gain increased visibility, collaboration opportunities, redundancy, and access to additional features and integrations. Additionally, while development and complex pipelines can remain within the private GÉANT GitLab environment, the final outputs can be publicly shared on GitHub.

The software may include non-original artefacts or assets with different licences. These assets, which may not be detected by SCA tools, should be documented with their origin, copyright, and licence information as soon as they are added to the project. Methods for documenting them are covered in Section 2.11 Licences and Tracking of Documentation, Data, and Other Works. Failing to document them promptly can complicate their identification and tracking in the future.

One Project or Several Projects?

The choice between a unified and a modular architecture is a trade-off between compliance control and operational efficiency. A single project simplifies licence management and ensures consistency across code, dependencies, and branding. It supports comprehensive analysis and reduces the risk of inconsistencies or legal issues. The downside is increased dependency volume and more demanding software composition and licensing analysis. A multi-project setup reduces dependency scope per component and allows more flexible development, distribution, and use, but requires managing compliance and licensing across multiple units. Use a unified approach when simplicity, consistency, and centralised compliance are required. Use a modular approach when flexibility, portability, and independent deployment are more important.

When handling multiple projects, it is essential to determine which dependencies should be included in the SCA analysis. This may depend on the relationship between components and their respective responsibilities. For instance, a project may be a subproject of the same team or a module within a larger project overseen by an external group or entity. Both the subproject and the main project may need to be analysed together with their source code and dependencies, even if stored in separate repositories.

This extended analysis is important for two reasons. First, an integrated analysis of the larger project and its subprojects provides a more comprehensive understanding of components. Second, an integrated view ensures that the larger project's licence governance is sound.

Even when contributions are perceived as part of another developer's project, contributors are not fully shielded from potential licensing disputes, especially if there is a formal relationship between the teams. To mitigate this risk, it is advisable to analyse the larger project as thoroughly as the contribution itself.

Developers should inform the licensing team of such situations and indicate which additional components should be included in the analysis. This can be facilitated by the configuration of the project's build tool. For example, Maven dependencies with the default "compile" scope are included in all build tasks and are propagated to all dependent projects, with their dependencies transitively included in the Mend analysis. Dependencies with "provided", "runtime", and "test" scopes, which are supplied by the execution environment at runtime or during testing, are excluded from the analysis, as they are considered external libraries.

In some cases, the development team may not yet have its own software and may consider using an external library or framework as the foundation for future development. The analysis of a non-GÉANT project with Mend for licences and vulnerabilities is a resource-intensive process and requires consultation with the licensing team on a case-by-case basis. Such analysis is conducted only when essential for a strategic decision.

2.6.3 Information Gathering and Documenting

Collect and document copyrights, licence findings, and decisions:

- **Specify the current licence** of your product (a packaged bundle of software, services, and components) or project (a programme or component), if one is already in place.
- Record the copyrights for the contributions used.
- **Request** software composition analysis (**SCA**) **and** software licence analysis (**SLA**) **via GÉANT Jira** [20].
- Scan the project codebase, producing an **inventory of components** and their licences **through the SCA service. T**he service sets up the SCA project, scans the codebase, and delivers an SCA report.
- **Interpret** the SCA outputs and findings.
- If ambiguities or complex situations arise, or advanced support is needed, proceed with the **SLA service for additional assistance**.
- Further analyse and **document dependencies** and licences.
- Based on the SCA report and any additional security or vulnerability reviews, **review and update the inventory of components**, identifying:
  - Vulnerable open source components that should be removed or replaced.
  - Outdated open source libraries that need updating.
  - Confirmed licences for the components used (in-licences).
- The review may also **clarify ambiguities** or doubts:
  - The SCA tool may not properly identify a licence or may report some licences as suspect or ambiguous.
  - Information about a licence may be false, unclear, or contradictory.
  - Some licences may be recognised by multiple names.
  - Some permissive licences, such as BSD or Artistic, have unnumbered variants or are sometimes edited by software authors.
  - The applicability of “or later” clauses may be unclear or wrongly declared by editing the original licence text.
- **Document all findings** through reports, UI displays, and data exports.
- **Consult the licensing team and the IPR Coordinator**, who will help determine the appropriate licence to apply. Tools such as Licensing Assistant [14] or choosealicense.com [15] can assist in this process.
- **Select an appropriate licence.** The document *Open Source Licences Used in GÉANT* [18] provides insights into OSS licences and their relationships. Reading it improves understanding of OSS licences and supports selection, but the final decision must be made with the IPR Coordinator after the SCA scan.
- **Document decisions.** Some licences may be further refined during remediation and decision-making, but the reasoning behind licence selection, detected issues, and how they were or will be addressed should be recorded. The IPR Coordinator’s reasoning and approval should also be documented.

For more information about the licence selection process and related recommendations, refer to the *OSS Licences and Licence Selection* guide [17]. GÉANT project participants can also consult the white paper *OSS Licences in GN4-3 and GN5-1 GÉANT Project: Current State and Recommendations* [39].

2.6.4 Remediation

This stage focuses on resolving licence conflicts and fulfilling obligations. It may require varying levels of effort from developers, ranging from minor adjustments to significant changes, such as removing or replacing dependencies, developing alternatives, or refactoring code. In some instances, further remediation may be required after subsequent SCA scans.

Remediation actions may include:

- **Selecting a more appropriate licence** for the product or project that is compatible with dependencies.
- Implementing quick fixes, such as:
  - Updating or replacing vulnerable open source components.
  - Updating outdated open source libraries where possible.
  - Investigating unclear component licences or seeking clarification or relicensing from authors.
  - Purchasing necessary proprietary software licences.
  - Choosing among dual-licence options for components.
- Identifying any remaining incompatible licences.
- **Deciding on actions** for components with such licences:
  - Removing non-essential components.
  - Replacing them with available alternatives.
  - Moving them to the server side, providing functionality via a central service.
  - Developing alternatives to meet the required functionality.
- Accepting certain risks.
- Internally documenting dependencies, their licences, the features they provide, the decisions made, and the remediation actions undertaken.
- Repeating the SCA and remediation steps, if necessary.

2.6.5 Creating Licence-Related Artefacts

To ensure OSS licence compliance, several key artefacts are required. The full details of these artefacts and the information to be included in them are covered in part 3. A step-by-step overview of the process is given below.

- Provide a LICENSE file.
  - Clearly state the selected OSS licence by placing a LICENSE file in the root directory.
  - Ensure the licence text matches the official version exactly. MIT, BSD, and ISC licences require copyright information to be included in the LICENSE file.
  - When a LICENSE file is present, header comments in source code files (with licence, copyright, and disclaimers) are generally unnecessary, except in special circumstances.
- **Declare licence options** if available and used.
  - Some licences offer options that must be explicitly stated.
  - Options may include accepting later versions of the licence or relicensing under specific licences endorsed by the original licence.
  - Licence options are typically declared in the README file.
- **Declare the licence** in project documentation, on the website, and in the repository UI.
  - Update project documentation to reflect the selected OSS licence.

    - Use available repository UI features to declare the licence.
    - If integrating with other systems or package managers, such as npm or PyPI, specify the licence using its SPDX identifier in the project's metadata file (e.g., package.json or pyproject.toml).
- Document the project licence in the GÉANT Software Catalogue and IP Register.
    - The Software Catalogue allows the licence to be declared on the project home page.
    - The IPR Coordinator manages the GÉANT IP Register.
- Provide a copyright notice.
    - Add a copyright notice for the project in a **COPYRIGHT file**, including funding information.
    - Include copyright information for third-party components in compliance with their licences, listing each component's name, year, and copyright holder.
- **Produce a README file** containing licence and copyright information.
    - Include project licensing information in the README file, specifying the licences of used components if necessary.
    - Provide instructions for accessing the full licence text if this is not included in the LICENSE file.
    - Briefly explain the licence's implications for users and contributors.
    - If licensing badges are available, add one to the README file to clearly communicate the project's licensing information.
- **Document code modifications** as required by the licence.
    - A history of changes may need to be documented, depending on the applied licence.
    - If the modified software or component has a CHANGELOG file or equivalent, extend it with details of your changes, preserving its format.
    - Check the licence text or summaries, such as those in *Open Source Licences Used in GÉANT* [18], to determine whether documenting code modifications is required.
- **Document dependencies**, their licences, notices, and copyright information.
    - List the licences of directly included dependencies in a dedicated file or project documentation, typically in the NOTICE file, along with each component's copyright and links to its licence text. The file may also include disclaimers and details on the project's licence, tools used, and IP used. The README can summarise this information in a more concise form.
    - In the AUTHORS file, individual contributors may be credited, funding acknowledged, and the relevant GÉANT work package or project task referenced.
    - For source code components in subfolders, store their licences, copyright, and notices within those folders.
- Document licence adherence, contribution, and updating.
    - Provide guidance for users and contributors on adhering to licensing requirements in the README, CONTRIBUTING, or CODE_OF_CONDUCT file. Examples include guidelines from FileSender [40] and Atom [41].
    - Outline contribution and copyright guidelines, even if external contributions are not expected.
- Prepare and apply a Contributor License Agreement (CLA) if necessary.
    - If the selected licence requires a CLA, establish one to define contribution terms and ensure mutual understanding.
    - Some licences offer suitable CLA forms.
    - Clearly outline the CLA process in the README and describe it in the CONTRIBUTING file to ensure legal clarity for all involved.
    - Place the CLA in a file named CONTRIBUTOR_LICENSE_AGREEMENT, CLA, or include it within broader contribution guidelines in the CONTRIBUTING file.

- Creating a Developer Certificate of Origin (DCO) is a lightweight alternative to a CLA. By adding a "Signed-off-by: Name <Email Address>" line in their Git commit, contributors self-certify that they agree to the DCO's terms, affirming that they are either the original authors or have the right to submit the code, which can then be used in the project under its licence.

- Establish a licence notification mechanism.
  - If the licence or contribution policy may change, implement a notification mechanism for contributors and users.
  - This may include prompts during builds, repository notifications, or updates via project communication channels.
  - A pop-up with licence and copyright change information is also an option.
- Once licence artefacts are complete, they should be assessed by the SLA team and the IPR Coordinator.
- Complete the SLA/SCA *Evaluation and Feedback Survey* [42].

2.6.6 Continuous Licence Management

The aim of continuous licence management is to **integrate licence oversight into the regular software development lifecycle**.

- Integrate licence-related checks into the build process.
  - Incorporate SCA and other licence-related tools into the build process. This can be partially automated by integrating dependency and licence checks into CI/CD toolchains.
  - The Mend service provided by GÉANT and other tools [36] can identify dependencies and their licences or verify compliance. Setting up an "allowed licences" policy in Mend ensures that licence violations are detected early.
  - Some tools, such as the License Maven Plugin [43], can generate a list of dependencies and their licences, download licence files, check, update, or remove licence headers in source files, and update or create the main project licence file.
  - Verify and review tool outputs, as they are not foolproof.
- Establish continuous monitoring and compliance checks.
  - Stay informed about changes to the chosen OSS licence.
  - Review the potential impact of any licence updates on the project.
  - Establish processes for continuous compliance checks, **repeating SCA and SLA as needed** for new dependencies or licences, to ensure that licensing obligations are met.
- Perform regular audits of licence-related artefacts.
  - Conduct regular audits to ensure that the project's licence-related artefacts are accurate and up to date.
  - Promptly resolve any discrepancies to maintain legal clarity and compliance, as delays are likely to complicate resolution.
- In complex licensing situations, **seek advice from legal@geant.org** to ensure proper interpretation and compliance.

GÉANT software best practice *BP-B.6: Manage Sideground IPR* [44] recommends addressing pre-existing and external IP early and periodically repeating the process.

2.7 Licensing and Tracking Documentation, Data and Other Assets

Software-related artefacts and assets distributed with software or stored in its repository are generally subjected to the same open source licence (depending on the licence compatibility). However, when such artefacts include data, technical documentation, configurations, and user manuals, for such materials that are not software artefacts, for example separate guidelines, presentations, training or promotional materials, it is advisable to use the Creative Commons Attribution (CC BY) or Attribution-NonCommercial (CC BY-NC) licences. The GNU Free Documentation License (GFDL) is also noteworthy. While data is sometimes licensed under OSS licences, datasets are more commonly licensed under Creative Commons and Open Data Commons.

Software should clearly document and attribute external data sources either in the software itself or in its documentation, ensuring transparency and compliance with data licensing or usage terms. If data comes with specific licence requirements or restrictions, the software licence may be impacted, particularly if data is hardcoded, integrated into data structures, forms part of default datasets, or is included in configuration or bootstrap scripts. This applies to all data provided with the software or used for its operation. Such data should be listed alongside dependencies and included in the software's licensing analysis. If proprietary or open data licences apply, compliance with their terms is required, and the data should be referenced at least in the NOTICE file. However, if the data is widely used public reference information, such as the ISO list of country codes, acknowledgement in the documentation or project artefacts is usually sufficient, and licence analysis is generally unnecessary. Even if external data is not explicitly credited, it should be documented, with explanations of how it can be updated. This also applies to data contained in external code libraries or modules.

If data is dynamically fetched from external services or APIs during initialisation or regular use, it must be clearly referenced in the README or NOTICE file and in the project documentation. Examples include maps, environmental and sensory information, and all presentations or embedding of data from external sources. The software may also store, aggregate, or process user-created or external data, such as collaborative content, usage information, logs, harvested data, personal details from authentication services, network state, topologies or traffic, and datasets for artificial intelligence training. Documentation should clarify how such data is used and guide administrators or users on how to subscribe to or access external sources, as registration with third-party services is often required. Ideally, software should support multiple data sources, reducing reliance on specific ones. Users should be made aware of available alternatives.

Processing external or user-created data may require user consent, be subject to the service's terms of use, or depend on agreements between the service provider and the external data source. These arrangements do not affect software licensing, but their implementation should be feasible using the software. Developers should ensure data security, design for personal data protection, and include features supporting arrangements such as user consent, cookie management, and the display of privacy and data policies and terms of use.

Most OSS licences include disclaimers of warranties and liability, meaning software authors are not legally responsible for malfunctions, damages, or misuse. Regardless of this, it is important for reputation reasons that the software is reliable and secure. To facilitate this, GÉANT offers security-focused code reviews using automated analysis and expert assessments [19], related training [45], and infrastructure-level security support [46].

The use or modification of externally developed work other than software can affect software licensing, especially when it involves database models, architectural designs, development frameworks, or code generated by tools. If such work is under a specific licence, the software must comply with its terms, which may include attribution, restrictions on commercial use, or obligations for derivative works. If the software incorporates external database models, frameworks, or generated content to a significant extent, the licensing terms may apply to the entire project. This is especially relevant when using copyleft OSS licences or non-OSS licences such as Creative Commons licences containing Non-Commercial (NC), No Derivatives (ND), or Share Alike (SA) clauses. For example, a database model may require modifications, extensions, and optimisations, making permission

for derivative works crucial. Therefore, it is critical to review the licences and terms of such work before starting significant development. These works are usually copyrighted and should be documented (e.g., in a NOTICE file) and included in the code repository, preferably in a dedicated folder, regardless of whether their original or modified versions are used.

Since many of these works are unlikely to be detected by SCA tools, it is important to document and communicate their use and licences as soon as they are incorporated. This also applies to non-original software-related artefacts, such as assets, configuration files, scripts, technical documentation, and user guides. If they are distributed with the software, they should be kept in its source code repository and licensed under the same terms. If numerous such works are included, they should be annotated with easily searchable and aggregable provenance and licensing details in the software repository, recorded in the project's Software Bill of Materials (SBOM) [47], or marked with extractable metadata and comments. These details include the place of use, origin, copyright, and licence. Failure to do so when adding such assets can make their identification and tracking difficult later. This is especially important for non-original graphical or UI assets, such as images, vector graphics, JavaScript code, or UI layouts, which may not be part of external components but could be used by them, and therefore easily overlooked.

Original assets distributed with the software should be kept under the same licence and do not need to be individually tracked.

2.8 Licence Options and Multi-Licensing

The applied licence may allow additional conditions or permissions to be added, clarifying how others can use, modify, and distribute the software. If the licence offers such options, they are explained in the licence text. Common software licence options that code owners may explicitly state include:

- **Permitting users to choose between the original licence version or any later version** – This allows users to choose between the original licence version or any later version approved by the licensor. Such relicensing can be interpreted as licence-endorsed open-ended multi-licensing. For example, a statement specifying that a particular GPL licence version is required is typically placed in the README file. You can use the full statement offered by the GPL: *"This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version"*, or simplify it to *"This software is licensed under GNU General Public License version 3 or any later version"*. If the licence notice ends with "version 3" or "version 3 only", only GPL 3.0 applies. If no version is specified, the recipient can choose any published version of the licence. The notice can also specify that a proxy can decide which future versions of the GPL can be used, although this option is rarely used. The same applies to the LGPL, with "GNU Lesser General Public License" replacing "GNU General Public License".

- **Relicensing under a different licence** – Some licences allow relicensing under a secondary licence endorsed by the original one or chosen by the licensor. For example, the EPL 2.0 notice stating "*This program and the accompanying materials are made available under the terms of the Eclipse Public License 2.0, which is available at https://www.eclipse.org/legal/epl-2.0/*" [48] means the software can only be used under EPL 2.0. However, a secondary licence may be introduced, allowing recipients to comply with either the EPL or the Secondary License, for example: "*This Source Code may also be made available under the following Secondary Licenses when the conditions for such availability set forth in the Eclipse Public License, v. 2.0 are satisfied: the GNU General Public License (GPL), version 3.*". Besides GPL 3.0, Eclipse allows the designation of GPL 2.0, their later versions, and versions with explicitly identified GPL exceptions or additional permissions, such as the Classpath Exception, as Secondary Licences. Any licence granting rights at least as broad as those under the EPL can be

declared as a secondary licence. Adding such a clause is effectively relicensing, and all copyright holders must agree to the change. Under MPL 2.0 [49], the GPL, LGPL, and AGPL are allowed as secondary licences by default, enabling users to relicense or distribute the code under those licences, unless the licensor explicitly opts out. To prohibit relicensing, the licensor must, in addition to the mandatory statement from Exhibit A, include the optional notice from Exhibit B: "*This Source Code Form is 'Incompatible With Secondary Licenses', as defined by the Mozilla Public License, v. 2.0.*"

- **Extending certain rights beyond standard terms** – Certain rights may be extended beyond the standard terms of the original licence, such as allowing commercial use without open sourcing modifications, providing a patent grant, or permitting the distribution of proprietary versions. These extensions should be clearly articulated, specifying that they supplement the original licence without replacing or altering its conditions. For example, Apache 2.0 states that the software is provided "AS IS" by default, but additional warranties can be agreed upon in writing.

- **Placing limitations on certain uses or modifications** – Some licences may allow the addition of conditions, such as restricting non-commercial use, requiring the availability of modified source code, or ensuring compatibility with the original version. Any limitations should be added carefully, ensuring compatibility with the original licence. For example, Apache 2.0 allows additional restrictive clauses, but licensors should avoid introducing restrictions that contradict or undermine open source principles.

- **Choosing the jurisdiction governing the licence** – With the EUPL, the licensor can choose the jurisdiction under which the licence is governed, specifying the applicable legal framework.

It should be noted that SCA tools cannot interpret options, additional rights, or conditions introduced in free text.

3 Complying with a Selected Licence

This section guides developers through the steps to be followed once a licence is selected. It covers preparing necessary files, performing compliance checks, and reviewing adherence with the licensing team. It also provides essential information for developers seeking to address licensing issues independently. Links to GÉANT-approved example files will be provided as they become available from software projects.

3.1 Licence Compliance Requirements

3.1.1 Legal Responsibilities

A software licence must be applied to any software not intended for internal use only. The selected licence depends on the licences of components used, and the copyright statement may vary depending on the institutions involved. Where the software is provided as a service (SaaS) there is no need to declare its licence unless one of the used components is under a network-protective copyleft licence.

Developers must comply with the chosen licence and ensure licence compatibility. Failure to do so constitutes a breach and could lead to legal challenges and financial loss.

Most licences require a copy of the licence to be included, typically in the LICENSE file located in the root folder. Although the LICENSE is often the first file developers look for to learn about the licence, simply providing it is not sufficient from a legal standpoint. Even with the LICENSE file in place and the software and its dependencies aligned with the selected licence, **the licence must be clearly stated** in the documentation, particularly in the README file, especially when using licence options or offering multiple licences. Some licences only require their name or URL to appear in the documentation, but including the full licence text in a dedicated file is standard. A clear statement declaring the chosen licence for the software is necessary. For instance, the GPL family specifies this requirement in the "How to Apply These Terms to Your New Programs" section at the end of the licence text. Apache 2.0 explains this in the "APPENDIX: How to apply the Apache License to your work." Licence statements are generally placed in the NOTICE and README files.

While licence and copyright information usually need not be included in every source file header, the licence and funding should be declared and placed in a manner that allows easy access. If the software has a website, the licences should also be stated there. The source code repository may also offer a method for specifying the licence. GitHub and GitLab provide features to declare the licence through the repository's user interface, and GitHub can automatically detect the licence from the LICENSE file in the root folder.

Software authors may choose to offer their work under several alternative licences through dual- or multi-licensing. However, when such software is used in another project, only one of the available licences should be applied, and it should be compatible with the project's licence. All component licences must be compatible with those offered for the main project.

The minimal set of typical documentation files in a software project usually includes a README for project information, a LICENSE for licensing details, and a CONTRIBUTING file for contribution guidelines where external contributions are accepted. The applied licence may also require a CHANGELOG or CHANGES file to track project

updates. These files may use Markdown, with names ending with ".md". If so, they can be edited using online Markdown editors such as Dillinger [50] or StackEdit [51].

The following sections detail compliance requirements, covering the placement of information, compliance with the licences of used code, README, COPYRIGHT, AUTHORS, NOTICE and CHANGELOG files, and copyright and licence notices in the source code (adding these, or at least licence notice in README ensures that a licence for your code is properly declared and others can re-use it).

Good templates and reference examples for well-designed files containing licence information, copyright, and other assets are provided in the repositories of some projects that underwent SLA, and a few external ones are available [52]. Each section includes a brief overview of the document type, its purpose, one or more templates, and relevant examples.

3.1.2 Placement of Licensing and Copyright

The README file serves as the primary entry point for users and is highlighted by platforms like GitHub, which display its content when the project's root folder is opened. It should summarise key information, including copyright and licence. Hyperlinks can direct users to other files and external resources, such as the full licence text, the project website, or funding information. Online platforms such as GitLab, GitHub, and the GÉANT Software Catalogue manage and display information such as the licence via their user interface. In some cases, registration in specialised registries, like research software registries or the WordPress plugin directory, may also be necessary. These platforms often extract relevant information automatically from files like README or COPYRIGHT and present it in the project profile or metadata.

Declaring and maintaining a copyright statement is essential, as it is a prerequisite for meaningful licensing. A copyright statement specifies the applicable years and identifies the copyright holder. For example, it may state that the software is copyrighted by GÉANT, NRENs, or other organisations.

- A COPYRIGHT file should start with this copyright statement, which may be followed by additional copyright statements for holders of embedded IP.
- The statement should also be included in the README, as this file contains key project information.
- The copyright statement should also be included in the NOTICE file, if present.
- If a MIT, BSD, or ISC licence is used, the LICENSE file contains a copyright placeholder that must be populated.
- All, or selected, source files may include copyright and licence statements.

When declaring a software licence, ensure the following:

- Include the full licence text in a dedicated LICENSE file in the root folder, as expected by convention.
- Although some licences require only the licence name or URL, providing the full text in LICENSE remains standard practice.
- Do not rely on LICENSE alone; it is insufficient from a legal standpoint. Provide an explicit written declaration that clearly links the software name with the licence. The software name may appear in the declaration or be stated in the title of a document or section containing it.
- State the licence clearly and concisely, especially in the README, and wherever licence options or multiple applicable licences are elaborated, typically in the NOTICE, if present.
- It is good practice to declare the licence in the project's metadata (where supported) and in external standalone documentation, the project website, or a wiki.
- Ensure the software and its dependencies comply with the terms of the selected licence.

- Follow the licence's own instructions for declaring the chosen licence (for example, the GPL section "How to Apply These Terms…" or the Apache 2.0 "APPENDIX").

Although the licence text may describe available licensing options, it is not usually modified to reflect the specific choices made in a project. The NOTICE file should also contain disclaimers, if needed. For multi-licensed projects, clearly list all licences, explain the terms of choice in the README and LICENSE, and include the full text of each licence (inline in LICENSE or in separate files) – see Section 3.1.4 Multi-Licensing and Composite Licensing. Always ensure the code repository's licence information reflects the applicable licence or licences.

Contributors are listed in the AUTHORS file, which names individuals or organisations contributing to the software, possibly categorising them (e.g., as contributing with code, documentation, or testing). The source code repository, if used regularly, tracks contributions and can provide information for the AUTHORS file. COPYRIGHT, NOTICE, or CONTRIBUTORS files may recognise the project team and individual contributors, but it is best to use the AUTHORS file for data on individuals, unless this is already documented elsewhere.

Funding information may appear in the README, NOTICE, AUTHORS, and COPYRIGHT files, and may include logos or links to sponsor or grant provider websites. It should clearly state whether funding has influenced the project's direction or goals, as is often the case with GÉANT developments. EC funding information is obligatory if the code was developed using EC funds. Funders should also be acknowledged in detailed documentation.

Contribution guidelines should be documented in the README or CONTRIBUTING files, specifying how contributors can engage with the project, submit changes (including committing, branching, testing, and releasing), and follow the project's coding standards and licence. They may include links to templates, coding style profiles or guidelines, and instructions for different types of contributions.

Software, its licence, and associated background and sideground IP should be recorded in the GÉANT IP Register by the licensing team and the IPR Coordinator.

Modules in subfolders may have their own licences. Each subfolder with a different licence from the main project should include a separate LICENSE file, along with the necessary attribution or copyright notices. This information is crucial for subprojects or folders differing from the main project. Other important information for subprojects should be provided in their respective folders, similar to the main project. The README, COPYRIGHT, and NOTICE files of unmodified components should remain unchanged.

3.1.3 Compliance with Licences of Third-Party Code

In addition to adhering to your project's licence requirements, it is essential to meet the obligations of licences governing dependencies or reused code, including copyright and patent rules. This requires ongoing attention throughout development. Key actions include:

- Extending README, COPYRIGHT, and NOTICE files to explicitly declare and credit dependencies or reused code, clearly stating their licences.
- Retaining existing licence and copyright files and notices to ensure full original documentation and compliance with the respective licences.
- Attributing and documenting any modifications to reused code, updating modification history and the contributor list, as necessary.
- Staying informed about updates to licences for used code, as they may impact compliance requirements and require adjustments.

Eclipse- and Mozilla-licensed dependencies (direct or transitive) require explicit reference in project documentation.

Using code under Apache 2.0 with a different licence for your project involves specific obligations:

- Including the original copyright notice.
- Providing a copy of the Apache licence.
- Describing significant changes made to the original code.
- Maintaining a NOTICE file with attribution notes (original or added).

3.1.4 Multi-Licensing and Composite Licensing

When a project is distributed under multiple licences, the LICENSE file should act as a navigational map. It must explicitly state whether the user has a choice, as in dual- or multi-licensing, or whether different parts of the code are subject to different terms (composite licensing: "The core is MIT, but the /tools directory is GPLv3"). This file must also explain the implications of each option, such as how one licence may allow proprietary derivative works, while another ensures that the user's software remains open source. A commonly used combination is Apache 2.0 and MIT, which offers maximum compatibility across open source and commercial projects, including GPL 2.0, as it is compatible with MIT.

Developers may choose to include the full text of every licence in the LICENSE file, clearly separated by headers. Alternatively, the main LICENSE file may provide a summary of the options and point to separate files containing the full legal texts. For example, the summary could read: "*This project is dual-licensed under the <Licence A, e.g. Apache 2.0 License> and the <Licence B, e.g. MIT License>. You may choose either licence, depending on your purposes [e.g. for proprietary integration or copyleft compatibility]. See LICENSE-APACHE and LICENSE-MIT for full texts.*"

3.1.5 Repository Hygiene (What Not to Include)

Besides creating and including appropriate artefacts, and maintaining neat code organisation, developers should also consider what not to include in the project repository. A repository should be regarded as a public stage where every commit becomes a permanent record. Committing binaries or generated files bloats the repository and obscures meaningful code changes. Therefore, it should contain only the source code required to build the project. Not to mention that an untidy repository may lead to the accidental or intentional inclusion of third-party assets, credentials, API keys, passwords, actual configurations, trade secrets, or other sensitive or proprietary information.

Do not include executables (.exe, .bin, .so, .dll), build artefacts (/build, /dist), archives (.zip, .jar, .apk, .war, .tar, .gz), or likely secrets (.p12, .key, .jks). Instead of storing compiled code in the repository, use the project's CI/CD pipeline to build such artefacts and publish them on a website or as release assets in a separate release registry. If the build requires external or third-party assets, it should retrieve them from external locations.

3.1.6 Use of AI Tools

An AI chatbot can be used to generate an initial version of a template-based artefact by copying the template into an editor, adapting it if needed, uploading it to the chatbot in use, and requesting it to populate the template using an existing artefact (online or attached), a project website, or details provided in the prompt. The generated text must be validated and refined as needed.

REMEMBER: using AI prompts can create copyright issues, always transform prompts creatively and verify AI-generated content. Also, AI tools are just tools, and it is the responsibility of the person(s) using them to verify the accuracy of their outputs.

3.1.7 Multi-Repository Projects

Developers of multi-repository projects may find it burdensome to repeat the same or similar information in compliance and documentation artefacts across all repositories, as this complicates maintenance. The following guidelines explain how to ensure navigability between repositories and minimise redundancy.

Mandatory Files for Each Repository

Every repository within a project, whether primary or subordinate, must contain the following files in its root directory:

- LICENSE: a complete copy of the project's licence.
- COPYRIGHT: a full statement, as in a single-repository project.
- README: a unique file tailored to the specific repository.

Even if the LICENSE and COPYRIGHT files in sub-repositories are identical to those in the primary repository, they must be physically present in each repository and cannot be omitted or replaced by links.

The README file for each sub-repository must provide a clear description of that specific component's purpose within the wider project. It must also include a succinct licensing declaration and a short copyright statement.

In this way, the sub-repository is self-contained and compliant with respect to licensing.

As a primary navigational tool, the README file must also include links to the main repository and other relevant sub-repositories. Project-level aspects, such as integrated installation, may be delegated to the README file of the main repository or a central component.

Sharing Files Across Repositories

While LICENSE, COPYRIGHT, and README files must be physically present in every repository, certain artefacts such as NOTICE, AUTHORS, CONTRIBUTING, CODE_OF_CONDUCT, and CHANGELOG may be centralised to reduce maintenance overhead and ensure consistency across the project. Subordinate repositories may contain only brief references or links to the authoritative version in the primary repository. For example, an AUTHORS file in a sub-repository may simply state that the full list of contributors is maintained at a specific URL in the primary repository.

A single, project-wide CHANGELOG may be maintained where repositories share a common release process and versioning scheme. Repository-specific CHANGELOG files should be considered only when individual components have independent release cycles or are maintained separately.

Here is an example of a sub-repository pointer file:

> # <Project Name> <Component> – <NOTICE/AUTHORS/CHANGELOG>
>
> This is a sub-repository of <Project Name>.
> The full <NOTICE/AUTHORS/CHANGELOG> for the entire project is maintained in [<Main Repository Name>](<Main Repository File URL>).

For projects not governed by the Apache licence, the NOTICE file may remain uniform across the project by placing repository-specific details, such as third-party component details or updates, in the relevant README or CHANGELOG files, and referring to the primary repository's NOTICE file for shared information. Apache-licensed projects must preserve and maintain NOTICE files by including specific information for all modified files, as required by the licence.

3.2 Artefacts (Project Files)

3.2.1 Mandatory, Conditional and Optional Files

For a legally robust foundation without unnecessary overhead, this guide defines only three files as mandatory: a LICENSE file specifying outbound legal terms, a COPYRIGHT file clarifying ownership and liability, and a README file providing essential open-source context and entry-point documentation. No other artefacts are required for baseline compliance; all remaining documentation is purpose- or context-dependent.

A NOTICE file and a CHANGELOG (or legacy CHANGES) file are recommended and may become mandatory where third-party dependencies or specific licence terms require attribution or modification tracking. Additional compliance artefacts, such as SBOMs (Software Bills of Materials) and REUSE metadata, may also become mandatory in the near future to support automated dependency tracking, vulnerability coordination, and licence automation. SECURITY documentation, a CONTRIBUTOR_LICENSE_AGREEMENT (CLA), or a DCO (Developer Certificate of Origin) may be introduced where required for security handling or inbound intellectual property management.

Community- and contributor-oriented governance files, including CONTRIBUTING, a CODE_OF_CONDUCT, AUTHORS, CONTRIBUTORS, and ACKNOWLEDGEMENTS, remain entirely optional and should be adopted as the project's collaboration model evolves.

A summary table of currently mandatory and conditional artefacts for GÉANT's software licensing certificates [21] is shown below.

Software Artefact	Self-Assessed Dependencies Certificate	Verified Dependencies Certificate	Verified Software Licence Certificate
README File	Optional (Early phase checklist/template)	Optional (Reviewed/amended by SLA Service)	Mandatory (Must include software info, licence, & copyright)
LICENSE File	Not Required	Not Required	Mandatory (Full text of selected software licence)
COPYRIGHT File	Not Required	Not Required	Mandatory (Explicit copyright ownership details)
NOTICE File	Conditional (Required only if mandated by dependencies)	Conditional (Required only if mandated by dependencies)	Conditional (Required only if mandated by dependencies)
CHANGELOG File	Not Mentioned	Not Mentioned	Conditional (Required only if mandated by licence/dependency)

Mandatory and conditional software artefacts

The contents of the individual files are detailed in the sections that follow.

3.2.2 README

The README file should contain basic information about the software, including the licence and copyright, typically in one short line each. It should also mention the origin or motivation for the development, and refer to the COPYRIGHT, LICENSE, and other files for details.

The README file should provide guidance and instructions related to the software, covering:

- Purpose or intent, which authors may sometimes omit as it may seem obvious to them, and key features.
- Scope, supported settings and environments, and requirements or constraints of the application, which may not be apparent to new users.
- Installation and configuration.
- Usage.
- Documentation.
- Roadmap and known issues.
- Community contributions.
- Funding, acknowledgements, dependencies, and tools used.
- Software **licence** and copyright.

The licence notice and details on licence options (such as “or later”), which may also be included in the NOTICE file, should be mentioned in the README. Markdown examples for stating the licence include:

```
## Licence: [GPLv3 or later](https://www.gnu.org/licenses/gpl-3.0.html)
```

or, in a slightly longer format:

```
## Licence
This project is licensed under the MIT License - see the [LICENSE](LICENSE) file for details.
```

or:

```
## Licence
This software is licensed under the GPLv3 or later. For more details, see the [LICENSE](LICENSE) file.
```

Since README files typically appear on repository home pages, the copyright statement and the EU emblem with the accompanying funding text should be included. Guidelines for formulating these are detailed in the COPYRIGHT section.

The list of direct dependencies in the README can be more concise than the one in the NOTICE. Both lists must be maintained. The README list may also include details on tools, libraries, and other software used, as illustrated in the following Markdown example:

```
## Dependencies
This project relies on the following third-party libraries and tools:
- [Library A](https://librarya.com) (MIT) - Used for data processing
- [Framework B](https://frameworkb.io) (Apache 2.0) - Provides core functionalities
- [Tool C](https://toolc.org) (GPL 2.0) - Assists in automating tasks
```

Listing frameworks and tools in the README helps describe the software's operational environment, supporting users, and contributors who may wish to reproduce the setup or use the same tools.

A helpful README template is available at *Make a README* [53], which provides exemplary Markdown and detailed recommendations in the section *Suggestions for a good README*.

3.2.3 COPYRIGHT

Software is protected by copyright law. Various OSS licences impose specific requirements under which software developers grant other users specific rights while retaining copyright. Therefore, developers must clearly indicate the copyright.

Copyright management within a project is expressed through a copyright statement, sometimes embedded in the LICENSE file. For short licences such as MIT, BSD, and ISC, the copyright statement is often part of the LICENSE file, typically placed at the beginning. In other cases, it appears in a separate COPYRIGHT file to preserve the integrity of the LICENSE file.

Not all software developed within the GÉANT project correctly indicates copyright, which must be addressed when working on software licensing. While core copyright information can be provided on the project website, in the LICENSE file, and in the README, it is advisable to create a dedicated COPYRIGHT file with more comprehensive details for software created or modified within GÉANT.

The COPYRIGHT file should be placed in the project's root folder, or in the component's root folder if different. For software that is GÉANT project output, it should reference the GÉANT project and specify the years during which the work was carried out, stating that the software is copyrighted by GÉANT and other contributing organisations. Additionally, it should indicate whether any code was reused, adapted, or derived from other sources. If contributions come from NRENs that were created independently of GÉANT, or include adapted code from other projects, the original copyright holders should be acknowledged, with their copyrights preserved if present.

Although the COPYRIGHT file does not need to declare a licence, it may, in addition to copyright information, include legal notices such as warranty and liability disclaimers, and/or acknowledgements of project funding and contributors, such as standard funding attribution texts and emblems. It may also include guidance on the treatment or transfer of copyright for new contributions, potentially with a reference to CONTRIBUTING.

For non-OSS software, the COPYRIGHT file must also assert "All rights reserved" and prohibit unauthorised use, copying, or modification unless a valid licence is granted.

If you are creating a COPYRIGHT file in Markdown format, Dillinger [50], StackEdit [51], or other online Markdown editors can be helpful. However, this is unnecessary if the copyright information is short and simple.

An example of a COPYRIGHT.md file with a disclaimer that can be adapted to your needs is provided below [the latest required EU funding emblems can be downloaded from the EC website here [54]:

> *"Copyright (c) 2025-2026 GÉANT Association on behalf of the GÉANT project*
>
> *THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES, OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT, OR OTHERWISE, ARISING FROM, OUT OF, OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.*

Project IP was generated and/or developed during the GÉANT project, a project that has received funding from the Horizon Europe research and innovation programme.

The Partner that developed the Project IP remains the sole owner of the Project IP developed during the Project. However, GÉANT Vereniging (Association), registered with the Chamber of Commerce in Amsterdam with registration number 40535155, and operating in the UK as a branch of GÉANT Vereniging (Association), registered office: Hoekenrode 3, 1102BR Amsterdam, The Netherlands, UK branch address: City House, 126-130 Hills Road, Cambridge CB2 1PQ, UK, has the free-of-charge, non-exclusive, perpetual, irrevocable, worldwide right to exploit the Project IP, including any Background and Sideground IP, and any IPR attached to the respective IP, including the right to sublicense the Project IP through multiple levels of sublicences and/or other licensing arrangements, and to release to third parties the Project IP, including Public Disclosure in accordance with the GÉANT IPR Policy."

The COPYRIGHT file begins with a copyright statement. It should follow the format "Copyright (c) <Year Range> <Copyright Holder>", where <Year Range> refers to the year or range of years during which the software was developed or updated, and <Copyright Holder> is the name of the organisation. For GÉANT, include the project phase in which the licensed software was developed, for example "Copyright (c) 2024-2025 GÉANT Association on behalf of the GÉANT project".

Additional copyright statements approved by the IPR Coordinator, such as those for contributions independently developed by other partners like NRENs, should be added after the GÉANT Association copyright. Contributors from GÉANT cannot claim copyright. Contact the IPR Coordinator with any questions about copyright.

The template includes a disclaimer of warranty and a limitation of liability clause. If the software is not open source and you wish to retain all rights under copyright law, include the phrase "All rights reserved" instead. Even without this statement, the work is automatically protected by copyright, meaning no one may reproduce, distribute, or adapt any part of the work without permission.

When the code has been developed with EC funding, the EU emblem and funding statement. For code developed within the GÉANT project and Horizon Europe programme, the "Co-funded by the European Union" emblem must be included. You can download the latest versions of the emblems and funding statements from the EC website here [54]. If you are unsure which EC emblem and funding statement you should use for your project, please contact techauthors@geant.org.

In addition to the COPYRIGHT file, the EU emblem and funding statement must be prominently displayed in the README, project documentation, the application's "About" page or pop-up, as well as in all public- or participant-facing communication materials, such as printed or digital products or websites. The same is recommended for NOTICE and AUTHORS, if present. Their use must follow the guidelines outlined in the *GÉANT IPR Policy* [3], summarised as follows:

- Minimum emblem height is 1 cm.
- It must include the phrase "Co-funded by the European Union".
    - The "Project IP was generated…" paragraph in the COPYRIGHT file should read "…was generated and/or developed during the GÉANT project, a project that has received funding from the Horizon 2020 research and innovation programme."
- Use a non-serif font without effects.
- The text should be proportionate to the emblem's size.
- The text should be in EU blue, black, or white, depending on the background.

- Ensure the text does not overlap with the emblem.

Figure 3.1 shows an example of the EU emblem with funding statement to be included as detailed above:

Figure 3.1: Example of EU emblem with funding statement

3.2.4 LICENSE

Always include the full licence text from its official source. To ensure licence detection by automated compliance tools, keep the text exactly as provided. The only permitted modifications are filling in any placeholders if the licence text includes them, such as copyright statements, as is the case with MIT, BSD, and ISC licences.

When only one licence is involved, provide supplementary information about licensing terms in the NOTICE file. For single-licence projects, keep the LICENSE file strictly for the legal text. To avoid modifying the LICENSE file and to keep the README file focused, supplementary information should reside in the NOTICE file.

When a project is distributed under multiple licences, the LICENSE file should act as a navigational map (see 3.1.4 Multi-Licensing and Composite Licensing for more info).

3.2.5 AUTHORS

Open source projects typically result from the collaborative efforts of various developers or teams, and it is important to acknowledge their contributions.

An AUTHORS (or CONTRIBUTORS) file is a simple way to credit individual contributors and reference the GÉANT work package or project task. This file should be placed in the project's root folder.

Funding acknowledgements may be included as required by the contract or programme. If individual contributor details are not provided, funding credits may be included in the COPYRIGHT file. Some information in the COPYRIGHT and AUTHORS files can also be summarised in the README file and the project's standalone website, as advised by the GÉANT Marcomms Team. GÉANT funding credits are not required for software descriptions on GÉANT-branded websites or internal collaboration platforms.

Here is a Markdown template for an AUTHORS file:

```
<Project Name> has been created by the GÉANT <GNx-y> WP<a> Task <b> <Recognisable Task Name>.
## Developers
- [<Author 1 Name>](<Author 1 Contact, hopefully working as a link>): <Author 1 Role>
- [<Author 2 Name>](<Author 2 Contact>): <Author 2 Role>
## Other Contributors
- [<Contributor 1 Name>](<Contributor 1 Contact>): <Contributor 1 Role>
- [<Contributor 2 Name>](<Contributor 2 Contact>): <Contributor 2 Role>
## Funding
- The [GÉANT project] (https://geant.org/projects/) is funded by the Horizon Europe research and
innovation programme.
```

To indicate the project's GÉANT origin, you may begin with the statement provided at the start of the above snippet.

When listing individuals, include their email addresses or other contact details (e.g., repository identifiers of contributors or ORCID identifiers), and a brief description of their contributions or roles, such as "Implemented the core functionality" or "Designed the user interface". This not only acknowledges contributors but also encourages new ones and facilitates collaboration. Be sure to include individuals significantly involved in the project, even if they did not directly author the code. Developers and contributors should be clearly distinguished from copyright holders listed in the COPYRIGHT file. You may also specify time periods, such as "2021:" and "2020 – present:". Ensure this section is regularly updated as new contributors join the project or make significant contributions.

After funding information, add other acknowledgements as needed. If the project is part of a larger ecosystem or depends on third-party libraries, it is good practice to acknowledge these dependencies either in the README or in a separate NOTICE file.

3.2.6 NOTICE

While the LICENSE file contains the broad legal terms of the project, the NOTICE file serves as the authoritative location for legal notices, attributions, and disclosures that must accompany the distributed software.

The project licence may require a NOTICE file, or, if modifying a project which already has such a file, additional entries to it may be required. For example, the Apache License 2.0 recommends including copyright information at the beginning of the NOTICE file, alongside the standard Apache License 2.0 boilerplate text that references the licence.

The NOTICE file should contain:

- Project identification and ownership: project name, copyright statement, primary licence with URL, and code repository location.
- Additional licensing details (if applicable): secondary licences, exceptions, or additional permissions.
- Standard warranty disclaimers: those in the LICENSE file may suffice, but inclusion is recommended.
- Derivative work note (if the project is a fork, derivative, or heavily based on another project): acknowledgement of the original project and its licensing requirements.
- Authors and contributors (optional): brief attribution or references to AUTHORS and CONTRIBUTORS files.
- Patent information (if applicable): patent grants or relevant patents owned by the organisation, particularly important for Apache 2.0 or GPL 3.0 projects.
- Trademark information: declarations of registered project trademarks and conditions of their use.
- Special acknowledgements and attributions (optional): licence-mandated, or provided for other reasons (contractual, policy, academic attribution, donations, community, upstream projects, moral rights, etc.).
- Funding statements (optional): funding source disclosures where required or relevant.
- Prior notices: preservation of, or references to, notices required by third-party licences.

The NOTICE file should also provide information about third-party components and other used IP in order to provide full transparency on the project's legal status. Furthermore, the Apache 2.0 licence explicitly requires using the NOTICE file for attributions. These external items should be explicitly listed:

- Third-party dependencies: name, version, origin URL, copyright notice, and licence, including required attributions.
- Third-party attributions for bundled assets or libraries included as source code: as above, plus preservation of existing notices and documentation artefacts in asset or library subfolders.
- General-purpose run-time components ("Platform Stack", optional): listed separately if significant.
- Tools used (optional): significant development or operational tools not part of the runtime.
- Trademarks (if applicable): if you mention or use third-party trademarks, for which you may need explicit permission, clearly state that these names and logos belong to their respective owners and are not covered by the software licence.

Below is a short Markdown example of a NOTICE file listing third-party components, their licences, URLs, and copyright information:

```
# <Project Name> - NOTICE File
This project includes third-party software components subject to open source licences. Please comply
with their licensing terms and attribution requirements:
1. <Dependency Name>
  - Version: <Dependency Version Number>
  - URL: <Dependency URL>
  - Licence: <Dependency Licence>
  - Copyright (c) <Dependency Year Range> <Dependency Copyright Holder>
We thank all open source developers who contributed to these dependencies.
```

The content and format of a NOTICE file may vary depending on the project and the licensing requirements of its dependencies. Licences often require a notice stating that <Project Name> is under the chosen licence, with information about applied licence options, secondary licences, exceptions, additional permissions, or disclaimers. Information about these possibilities is usually provided within or below the licence text. The offered boilerplate text typically needs to be adapted and placed at the start of the NOTICE file. Some licences require preserving content from the existing NOTICE files of dependencies.

If a NOTICE file is unnecessary, the above information can be provided at the end of the README file. It may also be repeated in the README file, omitting disclaimers and compacting the dependency list for brevity.

Using a SCA tool to scan the project helps identify overlooked dependencies. However, transitive dependencies listed in Mend reports (see Section 2.5.2 Mend SCA Tool) should not be included in the lists described above.

It is your responsibility to review and comply with the respective licences and attribution requirements for these components, as project dependencies and licence conditions evolve. Acknowledging contributions, dependencies, and tools fosters a collaborative open source community.

3.2.7 CHANGELOG

A well-maintained CHANGELOG enables users and developers to track a project's development history, making it easier to follow updates and modifications. It also aids in troubleshooting and identifying potential issues during upgrades. Even if not required by the licences, documenting the history of changes in the CHANGELOG is considered good practice. Missing CHANGELOG content may be reconstructed from repository versions and commits. Optionally, each version may include a short summary paragraph providing a high-level description of the release before the itemised changes. Such a summary may also be used where detailed items cannot be reconstructed.

Some licences require maintaining and providing a software change history, typically in a file named CHANGELOG, CHANGES, or HISTORY, located in the root directory. Certain licences have strict naming or content

requirements. The history should summarise new features, significant changes, additions, bug fixes, and other modifications chronologically, serving as a record of changes made to the software over its releases and over time. One common approach to generate the history is by concatenating release notes, which, in turn, can be derived from commit notes, release plans, and internal work documentation.

Entries should be organised with the latest changes at the top, each typically corresponding to a single release. Include release or version numbers or tags (e.g. "1.2.3"), as described in *Semantic Versioning* [55], and indicate releases or change dates. For unreleased changes, consider using "Unreleased" as the version until the release occurs. Summarise changes using bullet points or brief paragraphs, optionally grouping them by type under headings such as "Added", "Changed", "Deprecated", "Removed", "Fixed", and "Security". Alternatively, use indicative prefixes, such as "Added:", for each bullet point. Emphasise significant updates visually.

Systematic tracking of changes through commit notes is extremely helpful. Frequent commits, development in branches, and clear commit notes support this, but the information in the repository may be too specific. It is also important to tag releases with version numbers for easy identification. Other useful sources for tracking changes include release notes, lists of planned features and modifications, developer tasks, and fixed issues. The history should remain meaningful throughout the software's lifetime, so include the reason or context for changes, where useful. Change-related pull requests and code reviews may also provide additional context.

Use a consistent format for each entry, such as the following Markdown example:

```
## [1.2.3] – 2025-08-01
### Added
- New feature 1
- New feature 2
### Changed
- Update of an existing feature or its implementation
### Fixed
- Bug fix 1
- Bug fix 2
```

Version numbers in Markdown should link to the corresponding release tags. Descriptions of changes may reference relevant commit IDs and code files for more detailed information. Markdown linking to specific commits or issues related to changes provides access to further details:

```
- New feature 1 (#123)
- Bug fix 1 ([<Commit Hash>])
```

Developers should follow the recommended structure for each version entry and keep the CHANGELOG file up to date with each release. This results in clear, useful records that help users and developers understand the project's history and progress. The Common Changelog [56] outlines a suitable Markdown format and provides guidance for writing concise entries. Tools such as Keep a Changelog [57] can be integrated with the CI/CD pipeline to automate parts of this process.

The project documentation should explain that users and contributors can access the CHANGELOG file for release notes, and any other artefacts such as the project roadmap.

3.2.8 Source Code Notices (File Headers)

Placing copyright and licence notices in source code file headers ensures proper attribution and compliance with licence requirements and may improve transparency for contributors. However, this practice is declining due to

redundancy with centralised LICENSE files, maintenance complexity, interaction with version control systems and automation tools when the information is updated, and the preference for cleaner and more readable code.

A simple copyright line and licence indicator can be placed in the header of each source file, as shown below. While not mandatory, these notices are helpful if individual files are likely to be accessed, reused, or modified independently, and there are concerns that users or modifiers may overlook the project-level copyright and licence information.

```
// Copyright (c) <Year Range> <Copyright Holder>
// Licensed under the <Licence Name, e.g., “GPLv3 or later”>. See LICENSE file for details.
```

Rare exceptions where a copyright header is required are usually due to project licensing policies rather than licence requirements. For instance, Mozilla’s source repositories require appropriate header text for new code [58].

3.2.9 Project Files/Artefacts Checklist

This checklist provides key steps for preparing and validating project files, based on prior instructions and common issues:

- README
  - After the title with the software name, optionally list key attributes in bullet points.
  - Use subtitles to structure the content.
  - Briefly describe the software purpose, motivation, usage context, and prerequisites.
  - Provide brief installation, configuration, and usage instructions. Screenshots are encouraged. Include key notes on user management, integration, and security.
  - Clearly state the licence in one line, noting if options such as “or later” apply. Reference the LICENSE file or the official licence URL.
  - Include copyright in a single line or paragraph and refer to the COPYRIGHT file.
  - Mention the software’s origin or motivation.
  - Credit GÉANT and relevant parties, such as NRENs or external partners.
  - Include the EU logo and required text.
  - To keep the file short, refer to other files or documentation for further details.
- LICENSE
  - Confirm the licence with the GÉANT IPR Coordinator after ensuring component compatibility.
  - Copy the full licence text from the official source. Some licences include copyright placeholders to be filled.
- COPYRIGHT
  - Verify copyright and IPR details with the GÉANT IPR Coordinator.
  - Include the EU logo and required text. Host the logo as a static image in a controlled location.
- AUTHORS (optional)
  - List authors and contributors, including contact details (email, ORCID, or repository ID). Optionally include their institutions and roles.
  - Optionally credit the GÉANT project phase, work package, or task, while keeping copyright holders in the COPYRIGHT file.
- NOTICE (optional)
  - Not necessary if all licence-mandated information is in the README.

- Acknowledge third-party libraries, listing components, their licences, URLs, and copyrights.
- Optionally mention tools used in the project.
- Include mandatory notices for third-party components when required (e.g., for Apache 2.0 licensed components).

- CHANGELOG (recommended, often mandatory)
  - Maintain a concise history of changes. This is a requirement for many licences and is good practice.
  - Use a standard format.
  - For the first version, simply note it as the initial public or production release.

Refer to *Templates and Examples for Software Project Artefacts* [52] for guidance. A more detailed checklist is available with further instructions [59].

4 Resources

The following sections lists some helpful (both GÉANT and external) resources to assist developers in licence selection and compliance. A regularly updated and comprehensive list of such resources is kept by the SLM team at [16].

Some references listed below may be accessible only to GÉANT project participants and/or require authentication via eduGAIN [2] or other supported credentials.

4.1 Contacts

- IPR Coordinator email: legal@geant.org
- WP9 Task 2 Software Governance and Support, Software & Licence Management (SLM) – provides support for software licensing
  - Slack: **#sw-licences** in the https://geant-project.slack.com workspace
  - Email: sw-licences@software.geant.org

4.2 Training Materials

A list of GÉANT licensing courses and workshops is maintained by the software licensing team at [60].

Upcoming GÉANT training events and infoshares are posted in the GÉANT Events Management System [61].

4.3 Further Reading

- *GÉANT IPR Policy* [3]
- 'OSS Licences and Licence Selection' [17] – introductory guide
- *OSS Licences in GN4-3 and GN5-1 GÉANT Project: Current State and Recommendations* [39] – project-oriented white paper for GÉANT participants, providing guidance on licence selection, with an appendix describing the OSS licences most frequently used by analysed projects
- 'Software Licence Selection and Management in GÉANT' [62] – an online version of this guide maintained by the software licensing team
- 'Open Source Licences Used in GÉANT' [18] – descriptions, and compatibility and "cheat sheet" for commonly used licences in GÉANT
- 'Templates and Examples for Software Project Artefacts' (for GÉANT participants) [52] – per-artefact Markdown templates, and links to examples from various projects
- 'Reference Information about OSS Licences and Tools' [16] – a comprehensive catalogue of thematically organised resources and pointers
- GÉANT software best practice *BP-B.6:* 'Manage sideground IPR' [44]
- OSI Approved Licenses [12]

- ‘Software Artefacts Checklist’ [59] – essential guide for project compliance and required documentation
- ‘FAQ – Software Licensing Practices’ [63] – answers to common questions regarding GÉANT licensing
- ‘Glossary – Open Source Software and Licensing’ [64] – definitions of key terms

4.4 Services and Tools

GÉANT Licensing Services and Support

- GÉANT Software Licence Management (homepage) [65]
- Jira requests for SCA and SLA [20]
- ‘Software Reviews’ [19] – GÉANT Software Development Support
- GÉANT Mend (login via GÉANT SSO; special permissions may apply) [66]
- ‘Accessing Mend and visibility levels’ [67] – information on the visibility of Mend scan results
- ‘Mend short guide for end users’ [27] – includes interpretation of Mend reports
- ‘The Risk Report’ [68] – short end-user guide to the Mend risk report
- GÉANT GitLab [4][5] – the repository for GÉANT-related software
- GÉANT Software Catalogue [38] – the catalogue of GÉANT-related software
- SoftwareCertHub [69] – the central platform for viewing and managing GÉANT software licensing certificates
- Software Licensing Certificates [21] – guides to GÉANT software licensing certificates

Interactive selection and comparison tools

- Choosealicense.com [15] – simple, developer-friendly guidance for selecting the most common OSS licences.
- tl;drLegal [70]– provides plain-language summaries of what you can, must, and cannot do under specific licences.
- Licensing Assistant [14] – comparison of licences within the European legal context, with a focus on EUPL compatibility.

Software composition analysis (SCA) tools

- Mend Platform SCA (formerly WhiteSource) [26] – scans dependencies and provides risk-based assessment.
- ScanCode [71] and ScanCode Toolkit [72] – analysis of project artefacts for licences and credits.
- OSS Review Toolkit (ORT) [73] – for automated compliance checks.

Software bill of materials (SBOM) tools

- Trivy [74], Parlay [75], Syft [76], and Tern [77] – tools for generating SBOMs from containers and filesystems.
- CycloneDX Tool Center [78] – resources for securing the software supply chain.

Glossary

AGPL	GNU Affero General Public Licence
API	Application Programming Interface
BSD	Berkeley Source Distribution
CC	Creative Commons
CC BY	Creative Commons Attribution licence
CC BY-NC	Creative Commons Attribution-NonCommercial licence
CI	Continuous Integration
CI/CD	Continuous Integration / Continuous Delivery
CLA	Contributor License Agreement
DCO	Developer Certificate of Origin
EC	European Commission
EPL	Eclipse Public License
EU	European Union
EUPL	European Union Public Licence
EURISE	European Research Infrastructure Software Engineers
FAIR	Findability, Accessibility, Interoperability, and Reusability
GFDL	GNU Free Documentation License
GPL	GNU General Public License
ICT	Information and Communication Technology
IP	Intellectual Property
IPR	Intellectual Property Rights
ISC	Internet Software Consortium
ISO	International Organisation for Standardisation
LGPL	GNU Lesser General Public License
MIT	Massachusetts Institute of Technology
MPL	Mozilla Public License
NC	NonCommercial
ND	NoDerivatives
NREN	National Research and Education Network
ORCID	Open Researcher and Contributor ID
OSI	Open Source Initiative
OSLS	Open Source and Licence Support
OSS	Open Source Software
PLM	Product Lifecycle Management
R&E	Research and Education
SA	ShareAlike
SaaS	Software as a Service
SBOM	Software Bill of Materials
SCA	Software Composition Analysis
SLA	Software Licence Analysis
UI	User Interface
WP9	Work Package 9 Operations Support
WP9 Task 2	WP9 Task 2 Software Governance and Support

References

[1] CC BY 4.0 https://creativecommons.org/licenses/by/4.0/deed.en

[2] eduGAIN https://edugain.org/

[3] *GÉANT IPR Policy* https://resources.geant.org/publications/intellectual-property/

[4] GÉANT GitLab Community Edition (hosting most projects): https://gitlab.software.geant.org/public

[5] GÉANT GitLab Ultimate Edition (hosting selected projects, requires login): https://gitlab.geant.org/

[6] TinyMCE – Open Source Software Evaluation Checklist https://www.tiny.cloud/software-evaluation-criteria-checklist/

[7] Red Hat Checklist https://www.redhat.com/en/resources/open-source-project-health-checklist

[8] Software Quality Checklist https://technical-reference.readthedocs.io/en/latest/quality/software-checklist.html

[9] *GN5-1 Software Licence Selection and Management* https://resources.geant.org/wp-content/uploads/2024/04/GN5-1_Software-Licence-Selection-and-Management-in-GEANT.pdf

[10] *GN5-1 D9.4 Open Source and Licence Support Report* https://resources.geant.org/wp-content/uploads/2024/04/GN5-1_D9-4_Open-Source-and-Licence-Support-Report.pdf

[11] *GN5-2 D9.5 Open Source and Licence Support Report* https://resources.geant.org/wp-content/uploads/2025/11/GN5-2_D9.5_Open-Source-and-Licence-Support-Report.pdf

[12] Open Source Initiative approved licences https://opensource.org/license

[13] FSF FOSS Licenses https://www.gnu.org/licenses/license-list.html

[14] Licensing Assistant https://interoperable-europe.ec.europa.eu/collection/eupl/solution/licensing-assistant/find-and-compare-software-licenses

[15] Choose a License https://choosealicense.com/

[16] OSS Licences and Tools https://wiki.geant.org/spaces/GSD/pages/419758342/Reference+Information+about+OSS+Licences+and+Tools

[17] OSS Licences and Licence Selection https://wiki.geant.org/spaces/GSD/pages/1199931438/OSS+Licences+and+Licence+Selection

[18] Open Source Licences Used in GÉANT https://wiki.geant.org/spaces/GSD/pages/1036025888/Open+Source+Licences+Used+in+G%C3%89ANT

[19] Software Reviews https://wiki.geant.org/spaces/GSD/pages/1045004320/Software+Reviews

[20] Request a Software Review (requires login) https://jira.software.geant.org/servicedesk/customer/portal/2/create/55

[21] Software Licensing Certificates
https://wiki.geant.org/spaces/GSD/pages/1190199418/Software+Licensing+Certificates

[22] GÉANT Product Lifecycle Management (PLM) (requires login)
https://geantprojects.sharepoint.com/sites/plm

[23] Software Composition Analysis
https://wiki.geant.org/spaces/GSD/pages/1038811290/Software+Composition+Analysis

[24] Client Guide for Software Composition Analysis (SCA) (requires login)
https://wiki.geant.org/spaces/gn51wp9t2/pages/599785535/Client+Guide+for+Software+Composition+Analysis+SCA

[25] GÉANT Project Slack (requires login) https://geant-project.slack.com/

[26] Mend SCA https://www.mend.io/sca/

[27] Mend Short Guide for End Users
https://wiki.geant.org/spaces/GSD/pages/350584875/Mend+short+guide+for+end+users

[28] Adding a project to Mend Scan Flow
https://wiki.geant.org/spaces/GSD/pages/240844905/Adding+project+to+Mend+Scan+Flow

[29] Mend Risk Score Attribution https://docs.mend.io/legacy-sca/latest/risk-score-attribution-and-license-analysis

[30] GitHub https://github.com/

[31] GÉANT Public Bitbucket Repositories https://bitbucket.software.geant.org/repos?visibility=public

[32] GÉANT Bamboo (requires login) https://bamboo.software.geant.org/

[33] Automated Mend Scans with Bamboo
https://wiki.geant.org/spaces/GSD/pages/219938818/Automated+Mend+scans+with+Bamboo

[34] Hermine https://docs.hermine-foss.org/use_hermine.html

[35] Software Licence Analysis
https://wiki.geant.org/spaces/GSD/pages/1038811292/Software+Licence+Analysis

[36] Software Composition Analysis (SCA) and Software Inventory Tools
https://wiki.geant.org/spaces/GSD/pages/419758342/Reference+Information+about+OSS+Licences+and+Tools#ReferenceInformationaboutOSSLicencesandTools-SoftwareCompositionAnalysis(SCAandSoftwareInventory)Tools

[37] Open Source Licensing and Compliance course (requires login) https://e-academy.geant.org/moodle/enrol/index.php?id=214

[38] GÉANT Software Catalogue (requires login) https://sc.geant.org/

[39] *Open Source Software Licences in GN4-3 and GN5-1 GÉANT Project: Current State and Recommendations* (requires login) https://wiki.geant.org/download/attachments/633275197/GN5-1_Open-Source-Software-Licences-in-G%C3%89ANT.pdf?version=1&modificationDate=1683702764850&api=v2

[40] FileSender CONTRIBUTE file
https://github.com/filesender/filesender/blob/development/CONTRIBUTE.md

[41] Atom CONTRIBUTING file https://github.com/atom/atom/blob/master/CONTRIBUTING.md

[42] Evaluation and Feeback Survey (requires login)
https://wiki.geant.org/spaces/G52W9T2/pages/1107198188/V+Evaluation+Survey+-final

[43] License Maven Plugin https://github.com/mojohaus/license-maven-plugin

[44] Manage Sideground IPR Best Practice https://wiki.geant.org/spaces/GSD/pages/571932797/BP-B.6+Manage+sideground+IPR

[45] Secure Code Training https://wiki.geant.org/spaces/GSD/pages/192544799/Secure+Code+Training

[46] GÉANT Security https://security.geant.org/

[47] SBOM Standard Formats https://www.mend.io/blog/guide-to-standard-sbom-formats/

[48] Eclipse Public License 2.0 https://www.eclipse.org/legal/epl-2.0/

[49] Mozilla Public License 2.0 https://mozilla.org/MPL/2.0/

[50] Dillinger https://dillinger.io/

[51] StackEdit https://stackedit.io/

[52] Templates and Examples for Software Project Artefacts (requires login) https://wiki.geant.org/spaces/G52W9T2/pages/958103682/Templates+and+Examples+for+Software+Project+Artefacts

[53] Make a README https://www.makeareadme.com/

[54] EC download centre for visual elements (logos and funding statements) https://ec.europa.eu/regional_policy/information-sources/logo-download-center_en

[55] Semantic Versioning https://semver.org/

[56] Common Changelog https://common-changelog.org/

[57] Keep a Changelog https://keepachangelog.com/

[58] Mozilla License Headers https://www.mozilla.org/en-US/MPL/headers/

[59] Software Artefacts Checklist https://wiki.geant.org/spaces/GSD/pages/1032978728/Software+Artefacts+Checklist

[60] GÉANT Courses and Workshops https://wiki.geant.org/spaces/GSD/pages/419758342/Reference+Information+about+OSS+Licences+and+Tools#ReferenceInformationaboutOSSLicencesandTools-G%C3%89ANTCoursesandWorkshops

[61] GÉANT Events Management System https://events.geant.org/ (requires login)

[62] Software Licence Selection and Management in GÉANT https://wiki.geant.org/spaces/GSD/pages/1032978710/Software+Licence+Selection+and+Management+in+G%C3%89ANT

[63] FAQ – Software Licensing Practices https://wiki.geant.org/spaces/GSD/pages/1199931440/FAQ+%E2%80%93+Software+Licensing+Practices

[64] Glossary – Open Source Software and Licensing https://wiki.geant.org/spaces/GSD/pages/1265336493/Glossary+%E2%80%93+Open+Source+Software+and+Licensing

[65] GÉANT Software Licence Management https://wiki.geant.org/spaces/GSD/pages/1045004326/Software+Licence+Management

[66] GÉANT Mend (requires login) https://app-eu.whitesourcesoftware.com

[67] Accessing Mend and Visibility Levels (requires login) https://wiki.geant.org/spaces/gn51wp9t2/pages/599785530/Accessing+Mend+and+visibility+levels

[68] Mend Risk Report guide https://docs.mend.io/bundle/sca_user_guide/page/the_risk_report.html

[69] SoftwareCertHub https://certificates.software.geant.org/

[70] tl;drLegal https://tldrlegal.com/

[71] ScanCode https://aboutcode.org/scancode/

[72] ScanCode Toolkit https://github.com/nexB/scancode-toolkit

[73] OSS Review Toolkit (ORT) https://github.com/oss-review-toolkit/ort

[74] Trivy https://github.com/aquasecurity/trivy

[75] Parlay https://github.com/snyk/parlay

[76] Syft https://github.com/anchore/syft

[77] Tern https://github.com/tern-tools/tern

[78] CycloneDX Tool Center https://cyclonedx.org/tool-center/